

## **Anhang A - Glossar**

### ***Aufgabenangemessenheit***

Bei der Aufgabenangemessenheit geht es um die Frage, ob die Funktionen eines Systems (z.B. Maschine, Software oder Website) dazu dienen, die Ziele zu erreichen, die ein Nutzer damit verfolgt. Ein interaktives System ist dann aufgabenangemessen, wenn es den Benutzer unterstützt, seine Arbeitsaufgabe zu erledigen, d. h. wenn Funktionalität und Dialog auf den charakteristischen Eigenschaften der Arbeitsaufgabe basieren, anstatt auf der zur Aufgabenerledigung eingesetzten Technologie.

### ***Automatisierung***

Übertragung der menschlichen Arbeit auf Automaten unter Zuhilfenahme geeigneter Werkzeuge

### ***AVI***

Audio Video Interleave, ein von Microsoft definiertes Video-Containerformat. In einer AVI-Datei können mehrere Video-, Audio- und Text-Untertiteldatenströme vorhanden sein, die mit verschiedenen Verfahren kodiert sein können.

### ***Capturerechner***

Hier: Rechner, der Video – und Audiosignale über Videokarten empfängt und aufzeichnet. Auch als Videograbber bezeichnet.

### ***Effizienz***

Verhältnis vom Nutzen zu dem Aufwand, mit dem der Nutzen erzielt wird. Ein effizientes Verhalten führt wie auch ein effektives Verhalten zur Erzielung einer Wirkung, hält aber darüber hinaus den dafür notwendigen Aufwand möglichst gering.

### ***Elektroenzephalogramm***

Methode der medizinischen Diagnostik zur Messung der summierten elektrischen Aktivität des Gehirns durch Aufzeichnung der Spannungsschwankungen an der Kopfoberfläche. Ursache dieser Spannungsschwankungen sind physiologische Vorgänge innerhalb einzelner Gehirnzellen, die durch ihre elektrischen Zustandsänderungen zur Informationsverarbeitung des Gehirns beitragen.

### ***Elektrokardiogramm***

Das Elektrokardiogramm (EKG) ist die Registrierung der Summe der elektrischen Aktivitäten aller Herzmuskelfasern.

### ***Ergonomie***

Wissenschaft von der Gesetzmäßigkeit menschlicher Arbeit. Der Begriff setzt sich aus den griechischen Wörtern ergon (Arbeit, Werk) und nomos (Gesetz, Regel) zusammen. Zentrales Ziel der Ergonomie ist die Schaffung geeigneter Ausführungsbedingungen für die Arbeit des Menschen und die Nutzung technischer Einrichtungen und Werkzeuge, wobei neben der menschengerechten Gestaltung des Arbeitssystems vor allem die Verbesserung der Schnittstelle zwischen Mensch und Maschine eine besondere Bedeutung besitzt.

**Erwartungskonformität**

Ein Anwendungssystem ist dann erwartungskonform, wenn es den Anforderungen und Erwartungen, die ein Benutzer an das System stellt, entspricht.

**Eyetracker**

Gerät, mit dem man Blickbewegungen aufzeichnen und analysieren kann.

**Framerate**

Framerate ist die Anzahl Bilder, die von einer Kamera pro Sekunde aufgenommen werden.

**FTP**

File Transfer Protocol, ein Netzwerkprotokoll zur Dateiübertragung.

**HAW Hamburg**

Sofern nicht anders bezeichnet, ist damit die Hochschule für Angewandte Wissenschaften Hamburg gemeint.

**HDTV**

High Definition Television. Zeichnet sich gegenüber herkömmlichem Fernsehen durch eine erhöhte vertikale, horizontale und temporale Auflösung aus.

**Intention**

Absicht eines Menschen, etwas zu tun.

**Interaktion**

Wechselseitiges aufeinander wirken von Akteuren oder Systemen.

**Lernförderlichkeit**

Eine Anwendung ist lernförderlich, wenn es den Benutzer beim Erlernen des Systems unterstützt und anleitet.

**Metrik**

Eine Metrik ist eine (meist mathematische) Funktion, die eine Software zum Zwecke des Vergleichs oder der Bewertung in einen Zahlenwert abbildet. In dieser Arbeit sind Werte wie Mausklicks oder Tastaturanschläge gemeint.

**MFC**

Microsoft Foundation Classes. Eine riesige Klassenbibliothek zum Erstellen von dokumentbasierten Fensteranwendungen in der Programmiersprache C++.

**MPEG**

Moving Picture Experts Group ist eine Gruppe von Experten, die sich mit der Standardisierung von Videokompression und den dazugehörigen Bereichen, wie Audiodatenkompression oder Containerformaten, beschäftigt.

**Mousecapturer**

Anwendung, die Mausmetriken wie Mausklicks misst.

**Mousetracer**

Anwendung, die Mausbewegungen zu Auswertungszwecken grafisch darstellt.

**MVC**

Model – View – Controller, bezeichnet ein Architekturmuster zur Aufteilung von Softwaresystemen in die drei Einheiten: Datenmodell (Model), Präsentation (View) und Programmsteuerung (Controller). Ziel des Musters ist ein flexibles Programmdesign, das u. a. eine spätere Änderung oder Erweiterung erleichtern und eine Wiederverwendbarkeit der einzelnen Komponenten ermöglichen soll.

**Operator**

Assistent, der zumeist für die technische Bedienung zuständig ist.

**PAL**

Phase – Alternation – Line – Verfahren zur Farbübertragung beim analogen Fernsehen.

**Proband**

Gleichbedeutend mit Testperson.

**Public – Verzeichnis**

Ein Dateiverzeichnis, das öffentlich und somit für jedermann zugänglich ist.

**RAID 5**

Ein RAID – System dient zur Organisation mehrerer physischer Festplatten eines Computers zu einem logischen Laufwerk, das eine höhere Datensicherheit bei Ausfall einzelner Festplatten und einen größeren Datendurchsatz erlaubt als eine physische Platte. RAID 5 bietet sowohl gesteigerten Datendurchsatz beim Lesen von Daten als auch Redundanz, und gewährleistet Datensicherheit beim Ausfall von maximal einer Platte.

**Screenshot**

Abspeichern oder die Ausgabe des aktuellen graphischen Bildschirminhalts als Rastergrafik.

**Selbstbeschreibungsfähigkeit**

Eine Anwendung ist selbstbeschreibungsfähig, wenn jeder einzelne Dialogschritt durch Rückmeldung des Systems unmittelbar verständlich ist, selbsterklärend ist oder dem Benutzer auf Anfrage erklärt wird, so dass er nicht auf zusätzliche Hilfe angewiesen ist.

**Stream**

Kontinuierliche Übertragung von Daten.

**S – Video**

S – Video bezeichnet das getrennte Übertragen von Helligkeits – und Farbinformationen mit entsprechend ausgeführten Kabel – und Steckverbindungen.

**TCP/IP**

TCP/IP (Transmission Control Protocol/Internet Protocol) ist ein Netzwerkprotokoll und wird wegen seiner großen Bedeutung für das Internet auch kurz nur als Internetprotokoll bezeichnet. Die Identifizierung der Rechner geschieht über IP-Adressen.

**Usability**

Gebrauchstauglichkeit und Bedienerfreundlichkeit von Software.

**Videograbber**

Siehe Capturerechner.

**VirtualDub**

Freie Software zum Bearbeiten und Erstellen von Videodateien unter dem Windows – Betriebssystem.

**VLC**

VLC Media Player, ein freier Medienspieler und Streaming – Server.

**WMV**

Windows Media Video, ein neuartiger Videocodec, der Kopierschutzmassnahmen unterstützt.

**Workflow**

Synonym für Arbeitsschritt oder Arbeitsanweisung, die auszuführen ist.

**XML**

Extensible Markup Language (engl. für „erweiterbare Auszeichnungssprache“). XML ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien. XML wird u. a. für den Austausch von Daten zwischen unterschiedlichen IT-Systemen eingesetzt, speziell über das Internet.

**XviD**

Freier MPEG-4 – Videocodec.

## Anhang B - Bedienungsanleitungen

### B1 BuzzerTool

#### 1. Installation

Um das Buzzertool nutzen zu können, müssen zunächst einmal folgende Vorarbeiten getroffen werden:

##### **Schritt 1:**

Installation des Server-Jarfiles (BuzzerServer.jar) auf dem Testrechner.  
Näheres hierzu im folgenden Abschnitt.

##### **Schritt 2:**

Installation des Client-Jarfiles (BuzzerControl.jar) auf dem Testleiterrechner (Schnittrechner). Näheres im Kapitel 1.2.

##### **Schritt 3:**

Die Buzzerbox und das Mikrophon werden an den entsprechenden Schnittstellen des Testleiterrechners angeschlossen.

#### 1.1 Server-Jar

Um das Server-Jar zu installieren, kopiert man einfach das entsprechende Jarfile (BuzzerServer.jar) in ein beliebiges Verzeichnis.

Der Server kann danach über die Kommandozeile mit folgendem Aufruf gestartet werden:

```
java -jar %PATH%BuzzerServer.jar portnummer
```

%PATH% bezeichnet hierbei den aktuellen Pfad, in dem sich das Jarfile befindet.

Die Portnummer ist frei wählbar, jedoch sollten Reglementierungen wie z.B. Firewall beachten werden. Zudem muss die Portnummer mit der Portnummer im Client-Jar übereinstimmen, damit eine funktionierende Verbindung zwischen Probandenrechner und Testleiterrechner über das Buzzertool aufgebaut werden kann.

## **1.2 Client-Jar**

Um das Client-Jar installieren, kopiert man einfach das entsprechende Jarfile (BuzzerControl.jar) in ein beliebiges Verzeichnis.

Der Client kann danach über die Kommandozeile mit folgendem Aufruf gestartet werden:

```
java -djava.library.path=native\win32 -jar %PATH%BuzzerControl.jar
```

%PATH% bezeichnet hierbei den aktuellen Pfad, in dem sich das Jarfile befindet.

Damit die Anwendung gestartet werden kann, müssen folgende Änderungen durchgeführt werden:

1. Den Explorer öffnen
2. Im Menü auf Extras→Ordneroptionen→Dateitypen gehen, den Dateityp ‚JAR‘ markieren und Button Erweitert anklicken
3. Aktion ‚open‘ auswählen und auf Bearbeiten klicken
4. im Feld Anwendung für diesen Vorgang folgenden String einfügen:  
"%JAVA\_HOME%\bin\javaw.exe" "%2" -jar "%1" %\*

Jetzt ist es möglich, über eine Verknüpfung von BuzzerControl.jar die Applikation ohne Konsole zu starten.

5. Verknüpfung von BuzzerControl.jar erstellen
6. Eigenschaften der Verknüpfung auswählen und unter Ziel folgendes einfügen:

```
"%PFAD%\BuzzerControl.jar" -Djava.library.path=native\win32
```

Wobei %PFAD% der Pfad zum Verzeichnis ist, in dem sich BuzzerControl.jar befindet.

Weitere Installationen sind nicht notwendig.

## **2. Bedienungsanleitung**

Es folgt nun eine ausführliche Gebrauchsanweisung des Buzzertools.

### **2.1 Aufbau der Buzzerbox**

Die Buzzerbox besteht aus zwei grossen Buzzer sowie vier kleineren, farblich unterschiedenen Knöpfen, deren Funktionalität hier kurz angedeutet werden soll.

**Linker Buzzer** → Mit dem Buzzer auf der linken Seite lässt sich zu jeder Zeit ein Screenshot des gerade auf dem Testleitermonitors angezeigten Bildes machen.

**Rechter Buzzer** → Mit dem Buzzer auf der rechten Seite kann man zu jedem Zeitpunkt eine Sprachaufzeichnung bzgl. des Tests vornehmen, wenn etwas interessantes geschieht, das sich nicht nur allein per Bild festhalten lässt. Dazu bitte klar und deutlich in das angeschlossene Mikrofon sprechen. Die Aufzeichnung läuft, solange der Buzzer gedrückt ist.

Die bisher belegten Knöpfe sollen die spätere Auswertung des Tests erleichtern. Dazu wird am Ende des jeweiligen Tests eine Textdatei mit den gewonnenen Informationen automatisch generiert.

**Weisser Knopf** → Mit dem weissen Knopf (ganz rechts) werden alle erfolgreich bearbeiteten Aufgaben mitgezählt.

**Schwarzer Knopf** → Mit dem schwarzen Knopf werden grundsätzlich alle Aufgaben mitgezählt, egal, ob sie erfolgreich oder nicht erfolgreich bearbeitet werden konnten. Über den schwarzen Knopf wird auch der aktuelle Test gestartet.

**Roter Knopf** → "Reset-Knopf". Setzt alle internen Aufgabenzähler auf 0 zurück und schreibt die gesammelten Daten des letzten Tests über eine Textdatei auf die Festplatte. Mit dem roten Knopf wird das Testende signalisiert.

**Grüner Knopf** → Der grüne Knopf ist derzeit noch unbelegt, kann aber jederzeit mit weiteren Funktionalitäten ausgestattet werden.

## 2.2 Starten des Servers

Als erstes muss das Server-Jar über den schon beschriebenen Kommandozeilenbefehl gestartet werden.

Nach dem Start des Server-Jars, erscheint ein Konsolenfenster, wie in Abbildung B1 -1 dargestellt:

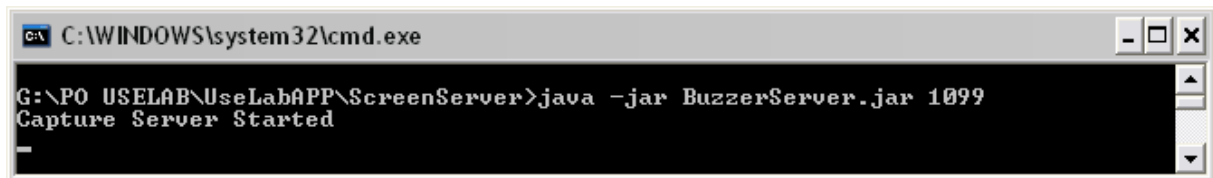


Abb.B1- 1: Starten des Buzzerservers

Somit ist der Capture-Server bereit.

Alle eventuell auftretenden Fehlermeldungen werden hier angezeigt.

Treten keine Fehler auf, kann der Client gestartet werden.

## 2.3 Starten des Clients

Der Client kann, wie oben schon beschrieben, direkt über die Verknüpfung gestartet werden. Nach dem Start des Client-Jars erscheint folgende Benutzerschnittstelle (Abbildung B1 - 2):

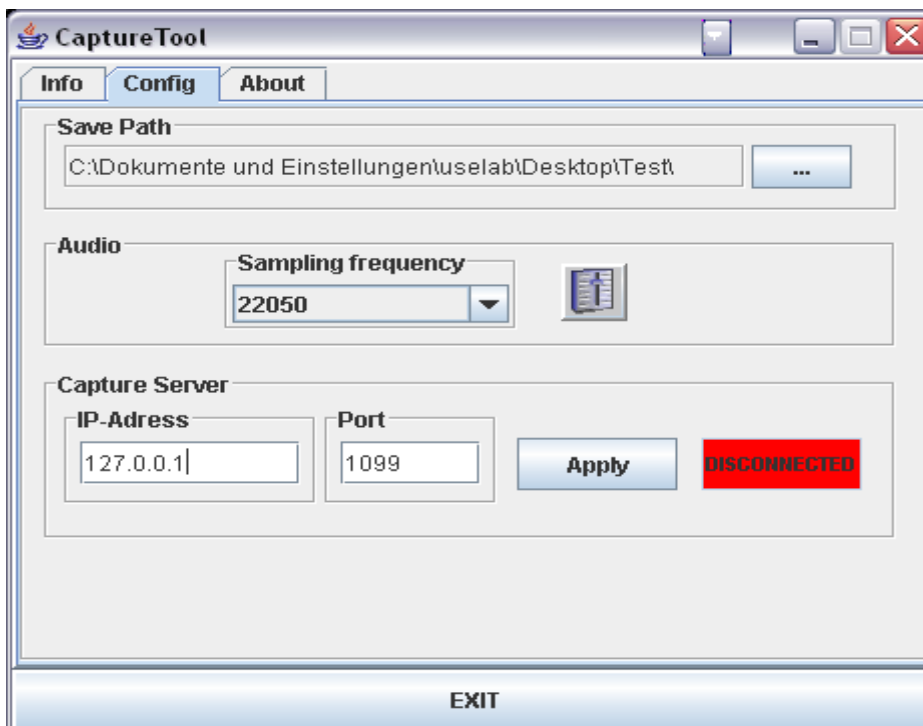


Abb. B1- 2: Starten des Buzzer-Clients

## **2.4 Einstellungen vornehmen**

Nachdem der Client erfolgreich gestartet werden konnte, sollten als nächstes einige Einstellungen vorgenommen werden. Die Menüseite dazu wird bei Programmstart automatisch angezeigt.

### **Save Path:**

Unter Save Path sollte ein existierendes Verzeichnis mit Schreibrechten ausgewählt werden, in dem später alle erzeugten Dateien (Screenshots, Textdateien, Audiodateien) abgelegt werden. Achtung: Ist der Pfad ungültig oder nicht mit den erforderlichen Rechten ausgestattet, können und werden keine Dateien gespeichert !

### **Audio:**

#### **→ Sampling frequency:**

Hier muss der entsprechende Aufnahmekanal für das Mikrophon ausgewählt werden (Möglichkeiten sind: 11025 kHz, 22050 kHz oder 44100 kHz). Eventuelle Lautstärkeanpassungen sind ebenfalls hier vorzunehmen. Dazu muss das Icon auf der rechten Seite angeklickt werden, welche einen Zugriff auf die Audiokonsole von Windows bietet.

### **Capture Server:**

#### **→ IP-Adress:**

Hier sollte die gültige IP-Adresse des Servers (Probandenrechner) eingegeben werden.

#### **→ Port:**

Hier muss der Port des Servers eingetragen werden. Der Port **muss** mit der Nummer, die beim Starten des Servers angegeben wurde, übereinstimmen.

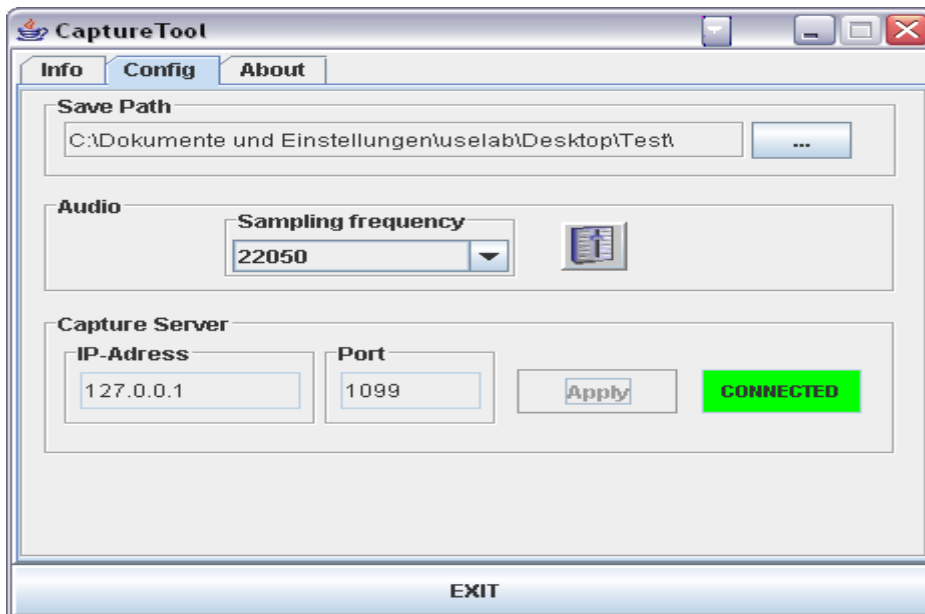


Abb. B1- 3: Der Capture-Client konnte erfolgreich mit dem Server verbunden werden

Sind alle Einstellungen vorgenommen worden, sollte mit einem Klick auf "Apply" die Verbindung zum Server hergestellt werden.

Falls der Server erreichbar ist, wechselt der Status der Anzeige (ganz rechts) auf Connected (grün), und das Buzzertool kann nun genutzt werden. Abbildung B1 – 3 zeigt einen erfolgreichen Verbindungsaufbau.

Falls der Server nicht erreichbar ist oder abstürzt, gibt es eine Fehlermeldung in der Benutzerschnittstelle, zu finden unter dem Tab "Info":

**ERROR GETTING SCREENSHOT!!!  
Please Check The IP And Port You Want  
To Connect To!!!**

Der Status des Capture-Servers wechselt von Connected auf Disconnected (rot). In diesem Fall bitte Server, IP-Adresse und Portnummer überprüfen.

### ***Die weiteren Tabs der Client-Anwendung:***

#### **Info**

Zeigt Informationen über alle Aktionen mit der Buzzerbox seit Programmstart an, wie in der nachstehenden Skizze (Abbildung B1 – 4) dargestellt.

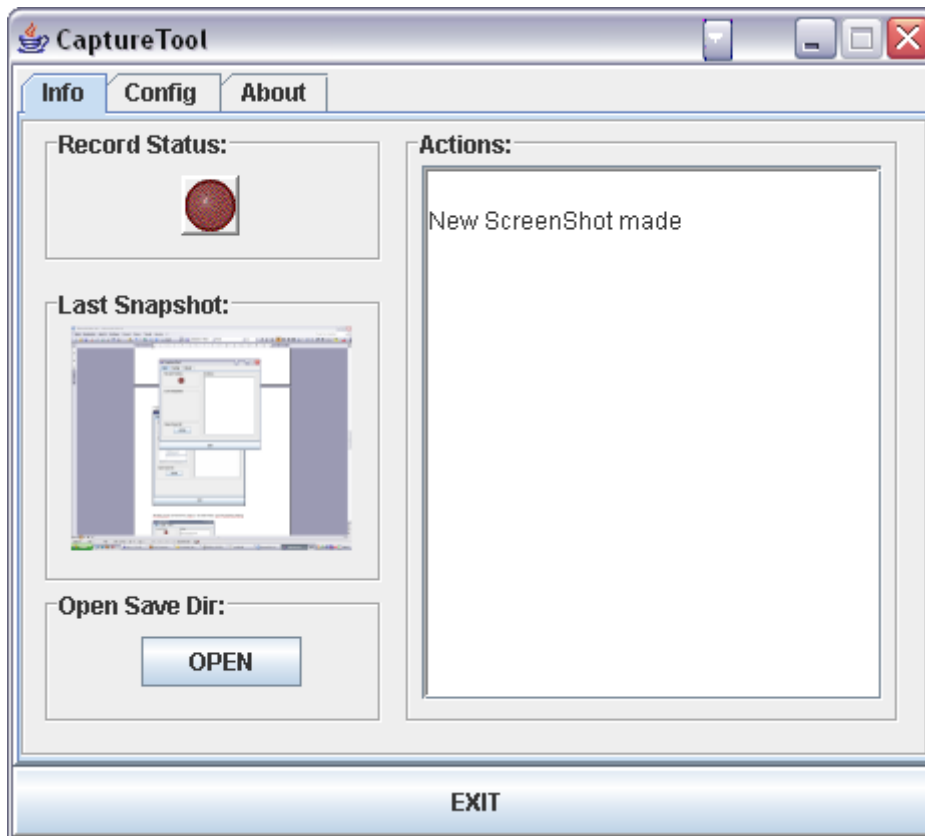


Abb. B1- 4: Das Feld Info

### **Record Status:**

Zeigt an, ob gerade eine Sprachaufzeichnung aufgenommen wird oder nicht.

### **Last Screenshot:**

Wenn ein Screenshot erzeugt wurde, so wird dieser hier in verkleinerter Form dargestellt. Ansonsten ist das Feld leer. Es wird immer der letzte Screenshot angezeigt.

### **Open Save Dir:**

Mit dem Button "Open" kann man den aktuell gesetzten Speicherpfad (der unter "Config → Save Path" gesetzt wurde) öffnen und anzeigen lassen. Die Dateien in diesem Ordner sind mit Datum und Zeit betitelt.

### **Actions:**

Einfache Textausgabe zu allen Aktionen mit der Buzzerbox seit Programmstart. Zeigt Meldungen über Beginn einer Sprachaufzeichnung, Ende einer Sprachaufzeichnung, Festhalten von Screenshots an, aber auch die aktuellen Zähler für Aufgaben gesamt und korrekt bearbeitete Aufgaben sowie eventuell auftretende Fehlermeldungen. Hier lassen sich alle Aktionen kontrollieren. Es wird empfohlen, nach Start des Clients und den vorgenommenen Einstellungen während des Tests diesen Tab geöffnet zu halten und das Feld "Actions" im Auge zu behalten.

## About

Informationsanzeigen über das Buzzertool und dessen Entwicklern.

→ Über den Button **EXIT** (zu sehen unter allen Tabs) kann der Capture-Server bzw. die Client-Applikation jederzeit beendet werden.

## 2.5 Festhalten eines Screenshots

Soll während des Tests ein Screenshot vom Bildschirm des Probandenrechners gemacht werden, braucht nur einmal kurz der linke Buzzer gedrückt werden.

Der Screenshot wird direkt im eingestellten Speicherpfad gespeichert und zusätzlich in der Anwendung verkleinert dargestellt. Abbildung B1 – 5 zeigt einen erfolgreich vorgenommenen Screenshot.

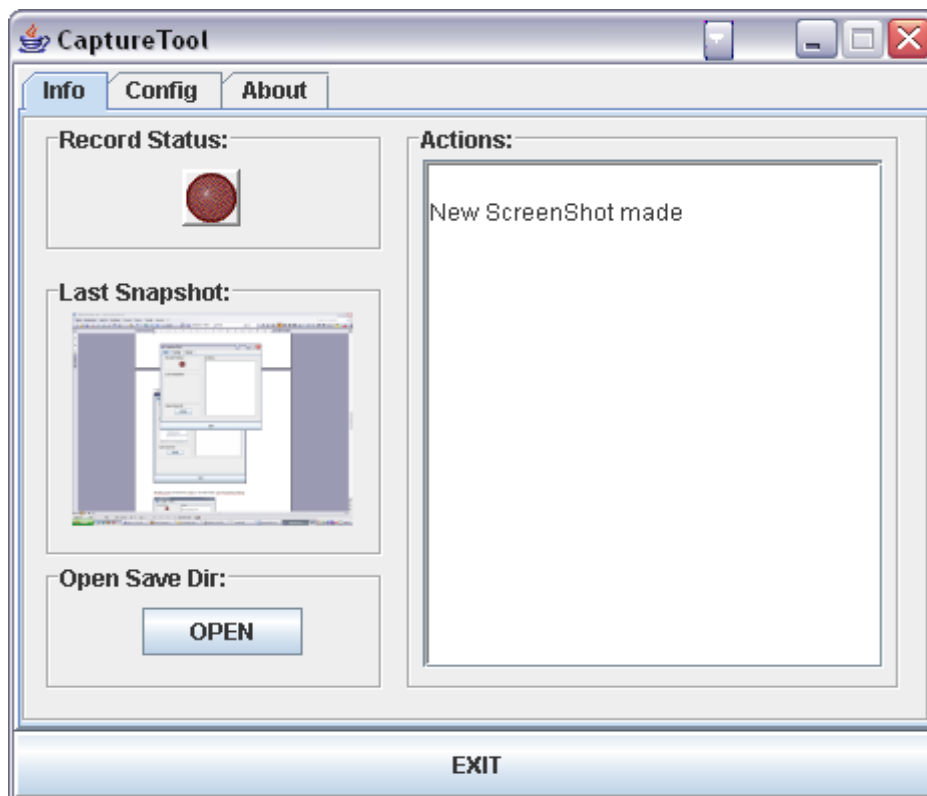


Abb. B1- 5: Screenshot war erfolgreich

**Achtung !!** Falls ein Zugriff auf das entsprechende Verzeichnis nicht möglich ist, erscheint folgende Fehlermeldung:

**ERROR SAVING SCREENSHOT!!!**  
**Do You Have The Rights To Access This**  
**Directory?**

## **2.6 Festhalten einer Sprachaufzeichnung**

Soll während des Tests eine Sprachaufzeichnung durchgeführt werden, muss der rechte Buzzer getätigt werden. Während der gesamten Aufzeichnung ist der rechte Buzzer gedrückt zu halten, wird der Buzzer losgelassen, so wird auch die Aufzeichnung gestoppt !

Nun kann man seine Beobachtungen über das Mikrofon festhalten.

Die Sprachaufzeichnung wird im WAV-Format innerhalb des Speicherverzeichnisses abgespeichert (bei Fehler gibt es eine Fehlermeldung, wie beim Screenshot).

Im Feld "Record Status" unter dem Tab "Info" kann man erkennen, wann eine Sprachaufzeichnung aktiv ist. Wird gerade aufgenommen, so wird anstellen des roten Kreises ein blinkender Kreis angezeigt.

Analog zum Screenshot hier 2 Abbildungen zum Audiorecording:

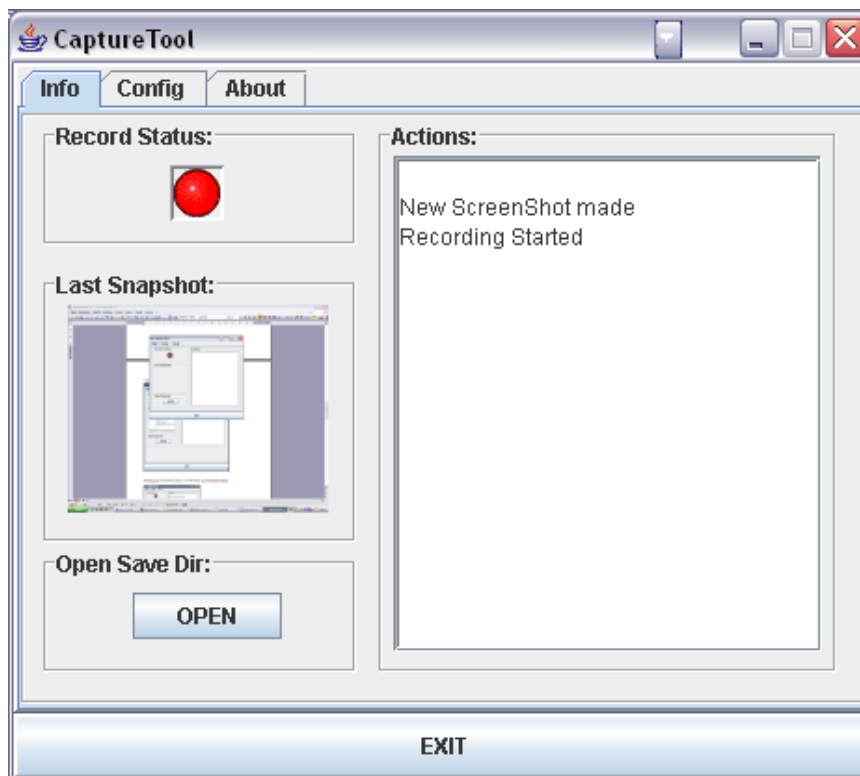


*Abb. B1- 6: Eine Aufnahme wurde gestartet*

Wie in Abbildung B1 – 6 dargestellt, wird durch den Eintrag „Recording Started“ im Feld Actions gemeldet, dass eine Sprachaufzeichnung gerade am Laufen ist. Abbildung B1 – 7 demonstriert entsprechend das Ende einer Aufzeichnung.

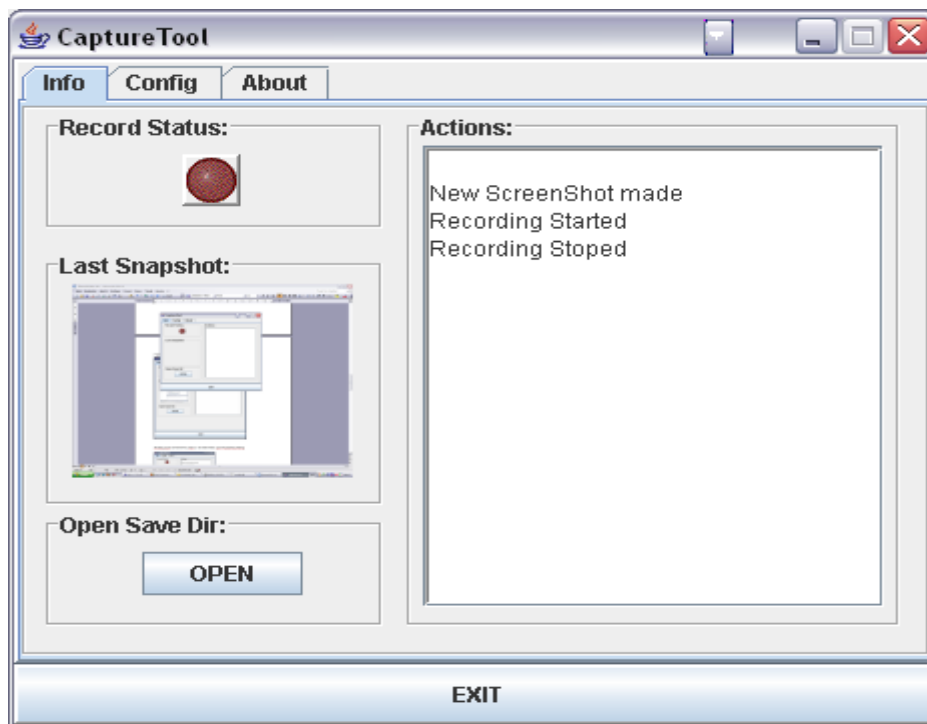


Abb. B1- 7: Die Sprachaufzeichnung wurde angehalten

## 2.7 Aufgabenzähler

Für den Aufgabenzähler werden drei von den vier kleineren Knöpfen der Buzzerbox benötigt: der schwarze, der weisse sowie der rote Knopf. Gezählt werden dabei die Aufgaben insgesamt (beginnend bei Aufgabe 1), die Anzahl der korrekt bearbeiteten Aufgaben (beginnend bei 1) – und damit indirekt auch die Anzahl der Aufgaben, an denen die Testpersonen gescheitert sind – sowie die benötigte Zeit für jede Aufgabe.

### **Achtung !**

**Es ist unbedingt notwendig, bei Gebrauch des Aufgabenzählers konzentriert mitzuarbeiten und wirklich nach jedem Anfang und Ende einer Aufgabe den entsprechenden Knopf zu betätigen, da sonst das Ergebnis verfälscht wird !!**

Die Ergebnisse werden zum einen über die Ausgabe "Actions" unter dem Tab "Info" (betrifft die Nr. der gerade laufenden Aufgabe, ob die letzte Aufgabe erfolgreich bearbeitet wurde oder nicht, und eine Information darüber, dass der Aufgabenzähler um eine Einheit hochgezählt wurde. Auch Fehlermeldungen bezüglich des Aufgabenzählers erscheinen hier), zum anderen mit allen vollständigen Informationen in einer Textdatei abgespeichert. Die Textdatei wird allerdings erst dann im Speicherverzeichnis abgelegt, wenn mit dem **roten Knopf** signalisiert wurde, dass der laufende Test beendet wurde.

Der **rote Knopf** hat weitere wichtige Besonderheiten, siehe entsprechenden Abschnitt.

Die einzelnen Funktionalitäten im Überblick:

### **Schwarzer Knopf: Gesamtaufgaben zählen**

Mit dem schwarzen Knopf werden alle Aufgaben, beginnend bei 1, gezählt. Hierbei ist es egal, ob die vorherige Aufgabe korrekt oder nicht korrekt bearbeitet wurde. Auch der interne Timer für die Dauer der jeweiligen Aufgabe wird mit dem schwarzen Knopf neu gestartet.

Es wird daher empfohlen, bei jeder neuen Aufgabe diesen Knopf zu drücken, da sonst versehentlich eine Aufgabe nicht mitgezählt wird.

Im Aktionsfeld (Tab Info → Actions) wird beim Druck auf diese Taste angezeigt, dass der Aufgabenzähler erhöht wurde und welche Aufgabe (in Form von Aufgabennummer) gestartet wurde. Man sollte diese Informationen im Auge behalten, um bei falscher Betätigung der drei Knöpfe entsprechend reagieren zu können.

### **Weisser Knopf: Korrekt bearbeitete Aufgaben zählen**

Der weisse Knopf verhält sich funktional ähnlich wie der schwarze Knopf. Allerdings sollte er nur gedrückt werden, wenn eine Aufgabe erfolgreich gelöst werden konnte.

**Bedingung:** Der Gesamtaufgabenzähler muss zuvor um eine Einheit erhöht worden sein.

Der Zähler für die korrekt bearbeiteten Aufgaben kann nur um 1 erhöht werden, wenn der Gesamtaufgabenzähler um 1 erhöht worden ist. Gleiches gilt für die allererste Aufgabe des Tests. Der weisse Knopf kann erst dann gedrückt werden, wenn zuvor die Aufgabe 1 mit dem schwarzen Knopf als "gestartet" gesetzt worden ist.

Ansonsten wird ein entsprechender Hinweis im Aktionsfeld dargestellt, dass zuvor der schwarze Knopf zu betätigen ist.

Wird der Richtig-Aufgabenzähler erfolgreich um eins erhöht, wird der interne Timer für die Aufgabendauer gestoppt und zurückgesetzt (die Dauer der Aufgabe kann später aus der angelegten Textdatei abgelesen werden).

Ausserdem zeigt das Aktionsfeld eine entsprechende Mitteilung an, dass die Aufgabe erfolgreich gelöst werden konnte.

### **Besonderes zwischen weisser und schwarzer Taste**

**Wird die schwarze Taste zweimal hintereinander gedrückt, ohne zwischendurch die weisse Taste zu betätigen, so wird die aktuelle Aufgabe als "nicht gelöst" gewertet und die nächste Aufgabe gestartet !**

Dies ist für die korrekte Handhabung der Tasten besonders zu beachten. Wurde also eine Aufgabe nicht gelöst, so muss keine zusätzliche Taste gedrückt werden, allerdings **darf** auch die weisse Taste dann **nicht** betätigt werden.

Die folgende Abbildung B1 – 8 zeigt die Ausgaben im Aktionsfeld, wenn

1. ein Fehler auftritt (weisse Taste gedrückt, ohne dass die Aufgabe mit schwarzer Taste freigegeben wurde)
2. eine Aufgabe korrekt erledigt wurde
3. eine Aufgabe nicht gelöst werden konnte.

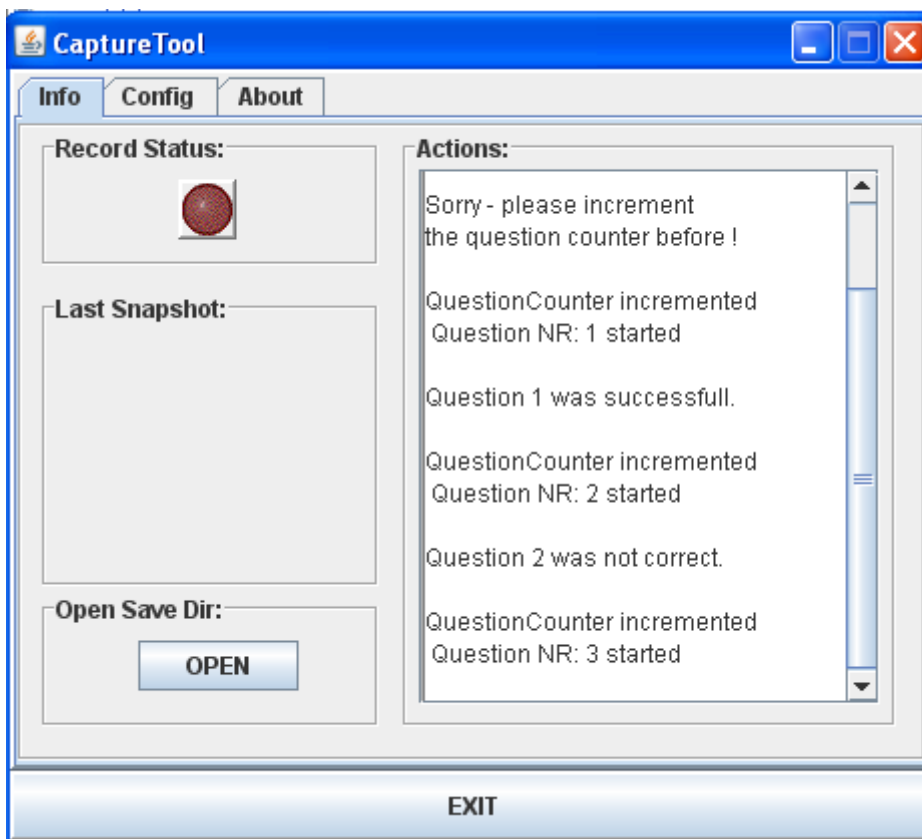


Abb. B1- 8: Schwarze und weisse Taste im Zusammenspiel – Ausgabe in der Anwendung

### **Roter Knopf: Zähler zurücksetzen und Textdatei speichern**

Mit dem roten Knopf wird der laufende Test beendet. Das heisst, die Textdatei mit allen Informationen wird im Speicherverzeichnis gespeichert und die beiden Zähler wieder in den Anfangszustand zurückgesetzt. Diese Taste ist nach Ablauf der letzten Aufgabe unmittelbar zu drücken.

Ein entsprechender Hinweis ist in dem Aktionsfeld ersichtlich. Ausserdem wird dort nun eine kleine Statistik (Anzahl Aufgaben gesamt und Anzahl korrekt bearbeiteter Aufgaben) über den Test gesamt angezeigt. Ein Beispiel ist der Abbildung B1 – 9 zu entnehmen.

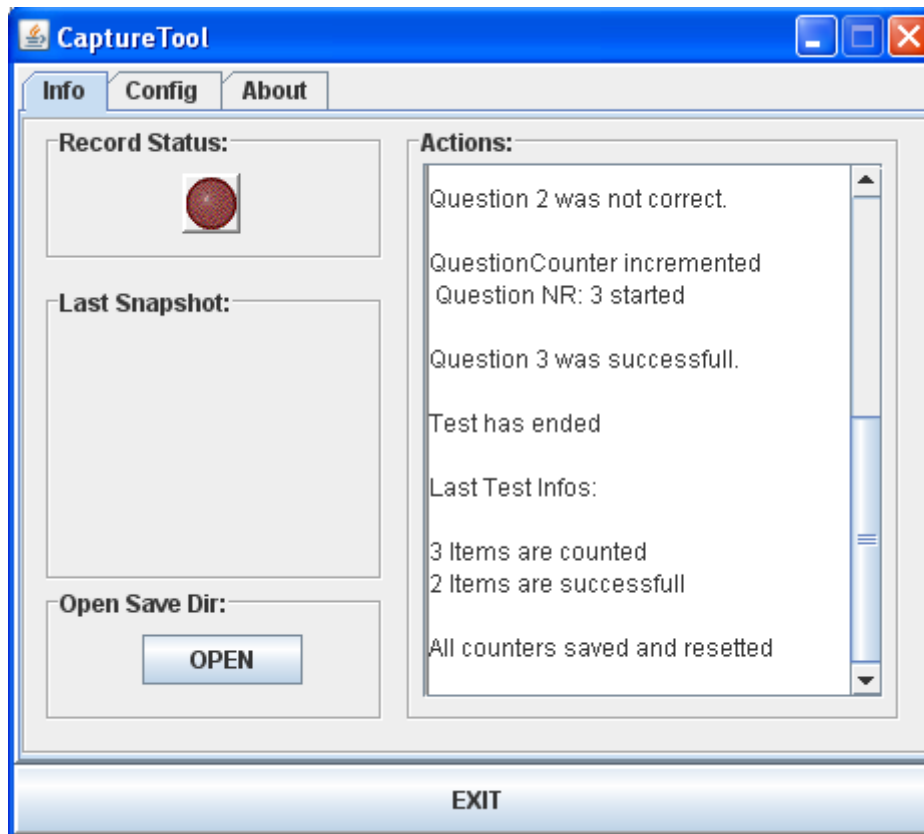


Abb. B1- 9: Die rote Taste wurde gedrückt und der aktuelle Test beendet

Im Speicherverzeichnis, welches als Speicherort eingestellt wurde, befindet sich nun ein Unterordner mit dem Namen Test+aktuelle Testnummer (z.B. Test1 für den ersten Testlauf seit Starten des Programms), in dem alle Dateien (Textdatei als ".txt", Sprachaufzeichnung als ".wav", Screenshots als ".png"), die zu diesem gerade abgelaufenen Test gehören, abgelegt sind. Man hat also am Ende alle zusammengehörigen Daten in einem einzigen separaten Ordner innerhalb des Speicherpfads zusammen.

Bei Betätigen des roten Knopfes wird ein weiterer interner Zähler des Buzzertools um 1 erhöht: der Testzähler. Das bedeutet, dass sofort der zweite Test beginnen kann, es wird dementsprechend für den nächsten Test ein neuer Ordner "Test2" im aktuellen Speicherverzeichnis angelegt !

Die vom Tool generierte Textdatei hat folgendes Aussehen (Abbildung B1 - 10):

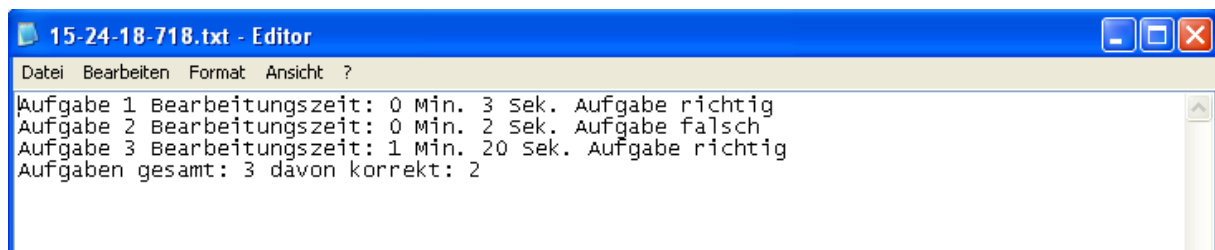


Abb. B1- 10: Die Textdatei zu dem letzten Test

Für jede Aufgabe existiert eine eigene Zeile, bestehend aus Aufgabennummer, Dauer für die Erledigung der Aufgabe sowie eine Angabe über Erfolg oder Nichterfolg bzgl. des Lösens der jeweiligen Aufgabe. Die letzte Zeile zeigt an, wie viele Aufgaben es insgesamt im Testdurchlauf gab, und wie viele von denen erfolgreich bearbeitet werden konnten.

Diese Textdatei kann auch als Eingabe für Auswertungstools wie das von Daniel Wolf genutzt werden.

### **3. Hinweise zur Programmierung**

Es folgt eine Übersicht über die verwendeten Klassen sowie jeweils eine kurze Beschreibung dieser Klassen und der darin enthaltenen Methoden. Hinweise für den weiteren Ausbau des Buzzertools (**grüner Knopf**) werden an den entsprechenden Stellen ebenfalls gegeben.

Die Realisierung des Buzzertools wird aus dem Grund auf diese Weise beschrieben, da hier noch Möglichkeiten zur Erweiterung vorhanden sind und diese Beschreibung gleichzeitig eine Dokumentation für die Entwickler sein soll, die sich zukünftig möglicherweise noch mit dem Tool programmtechnisch beschäftigen werden.

#### ***Package control***

Das Package control stellt den Controller im Sinne des MVC dar. Hier werden sowohl Server als auch Client gestartet.

Klassen:

#### **CaptureServer**

Die Klasse CaptureServer initialisiert den Server (Probandenrechner) und stellt den Dienst, Screenshots über TCP/IP zu versenden, zur Verfügung. Die Screenshots werden von dem Bildschirm des Rechners, auf dem der Capture-Server läuft, gemacht.

Methoden:

#### ***public static void main(String argv[])***

Startet das Programm zum Initialisieren des Servers (Aufruf von **BuzzerServer** über die Konsole und nimmt als Parameter die in der Konsole eingegebene Portnummer entgegen.

#### ***public CaptureServer(int port)***

Initialisiert das Server-Objekt, ruft den Thread für den Server auf (indem ein neues ServerSlave-Objekt der inneren Klasse angelegt wird, siehe weiter unten) und initialisiert ein Screenshot-Objekt zum Testen der Verbindung zwischen Client und Server.

***public static CaptureServer RUN\_SERVER(int port)***

Startet den Server, eine entsprechende Mitteilung wird auf der Konsole ausgegeben.

Die Klasse CaptureServer enthält eine innere Klasse ServerSlave, die für den Thread des Servers verantwortlich ist und von der JAVA-Klasse Thread abgeleitet wird. Diese innere Klasse enthält folgende Methoden:

***public ServerSlave()***

Startet den Server-Thread (wird vom Konstruktor der CaptureServer-Klasse aufgerufen).

***public void run()***

Wartet auf eine Anfrage des Clients. Erreicht eine Client-Anfrage den Server, wird ein Screenshot gemacht und zum Client zurückgesendet. Dazu muss eine Bytekonvertierung des Screenshots vorgenommen. Dieses dient nur zum Testen der Verbindung zwischen Client und Server.

***private void initStreamsAndSocket()***

Hier werden die nötigen Streams für den Server und den Client initialisiert. Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

***private void openSocket()***

Öffnet den Port für die Clients. Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

***private void closeStreams()***

Schliesst die Input- und Output-Streams. Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

***private void writeMessageToClient(byte[] bytelnmg)***

Sendet den nach Byte konvertierten Screenshot an den Client. Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

***private String getMessageFromClient()***

Methode zum Empfangen von Nachrichten, die vom Client an den Server gesendet werden. Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

***private byte[] toByteArray(BufferedImage image)***

konvertiert ein Bildobjekt (Screenshot) in ein Byte-Array.

**CaptureClient**

Die Klasse CaptureClient ist zuständig für die Anfragen an den CaptureServer.

*Methoden:*

***public CaptureClient(InetAddress ip, int port)***

Initialisiert das Client-Objekt und verbindet dieses mit dem Server.

***public void getScreenshot()***

Liest den empfangenen Screenshot, wandelt es in ein Bildobjekt zurück und speichert das Bild. Entsprechende Methoden werden hier aufgerufen.

***private void testConnection()***

Testet die Client-Server-Verbindung, indem die benötigten Streams initialisiert werden und eine Anfrage nach einem Screenshot (Testbild) gesendet wird.

***private void sendScreenShotRequestToServer(String mess)***

Sendet eine Anfrage nach einem Screenshot an den Server. Der Server sollte daraufhin ein Byte-Array zurückliefern. Schlägt die Anfrage fehl (z.B. wenn der Server nicht erreichbar ist), wird eine entsprechende Fehlermeldung ausgegeben.

***protected void initSocket(InetAddress ip, int port)***

Initialisiert den nötigen Socket für die Verbindung mit dem Server.

***protected void closeSocket()***

Schliesst den Socket, wenn er nicht mehr benötigt wird.

***protected void initStreams()***

Initialisiert die nötigen Streams für den Server und den Client.

Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

***private BufferedImage fromByteArray(byte[] imagebytes)***

Konvertiert ein empfangenes Byte-Array zurück in ein Bildobjekt.

***public void saveImage(BufferedImage image)***

Speichert das empfangene Bild. Die Screenshots bekommen einen speziellen Dateinamen, der sich aus aktuellen Uhrzeit (im Millisekunden) – hierfür wird ein Hilfsobjekt der Klasse UseLabUtilities benötigt, siehe entsprechenden Abschnitt – sowie dem Typ des Bilds (".png") zusammensetzt. Gespeichert wird das Bild in dem vom Anwender eingestellten Speicherpfad.

***public void setPath(String pa)***

setzt den Pfad, in dem Dateien zu speichern sind.

***public String getPath()***

Gibt den Speicherpfad zurück.

***public String getLastPath()***

Gibt den gesamten Speicherpfad inklusive des kompletten Dateinamen der zu speichernden Datei zurück.

**PollControl**

Ein Objekt der Klasse PollControl stellt den ‚Controller‘ der Hauptapplikation dar.

Hier werden im Sinne von MVC die erforderlichen Schritte eingeleitet, wenn irgendwelche Buzzer oder Knöpfe der Buzzerbox getätigt werden.

*Behandlungsroutinen für den noch offenen grünen Knopf sollten ebenfalls*

*aus dieser Klasse heraus aufgerufen werden.*

*Methoden:*

***public PollControl()***

Konstruktor des Controllers. Hier werden die benötigten Objekte angelegt und initialisiert (*für den grünen Knopf die entsprechenden Objekte hier ebenfalls anlegen*), die Listener gesetzt (wegen MVC) und Standardeinstellungen für die Benutzeroberfläche (View) bereitgestellt.

***public static void main(String[] args)***

Startet die Client-Anwendung.

***private void initClient()***

Initialisiert den Client und testet die Verbindung zum Server.

***private void setPaths()***

Ermittelt über den View den an der Benutzeroberfläche eingestellten Speicherpfad und gibt diese an das Recorder-, das Screenshot- und das Zähler-Objekt weiter. *Werden für den grünen Knopf weitere Objekte benötigt, sollte auch für diese der Speicherpfad in dieser Methode gesetzt werden.*

***public void makeScreenshot()***

Veranlasst das Screenshot-Objekt des Models dazu, einen neuen Screenshot vom Probandenrechner zu machen. Dabei wird überprüft, ob der Probandenrechner noch über das Netzwerk zu erreichen ist oder der Speicherpfad zum Speichern des Screenshots gültig ist. Ist dies nicht der Fall, werden entsprechende Fehlerbehandlungsmethoden aufgerufen.

***public void startAudio()***

Sendet die benötigten Angaben (unter anderem eingestellte Kanäle und Frequenzen) an das Recorder-Objekt und den View und veranlasst das Recorder-Objekt, die Sprachaufzeichnung zu starten.

***public void stopAudio()***

Veranlasst das Recorder-Objekt dazu, die Sprachaufzeichnung anzuhalten. Entsprechende Meldungen werden an den View weitergegeben, um das Stoppen der Aufzeichnung textlich darzustellen.

***public void incrementSuccess()***

Routine für den Fall, dass der weisse Schalter an der Buzzerbox getätigt wird. Wenn der Aufgabenzähler gesetzt und noch nicht der weisse Knopf gedrückt wurde, dann wird das Zählerobjekt dazu veranlasst, den Zähler für erfolgreich bearbeitete Aufgaben zu inkrementieren. Ansonsten wird an der Benutzeroberfläche eine Fehlermeldung ausgegeben.

### ***public void incrementQuestions()***

Routine für den Fall, dass der schwarze Schalter an der Buzzerbox getätigt wird. Der interne Aufgabenzähler wird über das Zähler-Objekt inkrementiert und eine entsprechende Mitteilung an der Benutzeroberfläche ausgegeben. Wenn vorher nicht der weisse Knopf gedrückt wurde, der Aufgabenzähler aber bereits gesetzt ist ( $>0$ ), dann wird an der Benutzeroberfläche eine Meldung angezeigt, dass die zuletzt bearbeitete Aufgabe als nicht korrekt bewertet wird.

### ***public void reset()***

Routine für den Fall, dass der rote Schalter an der Buzzerbox getätigt wird. Veranlasst das View-Objekt dazu, eine kleine Statistik über den abgelaufenen Test an der Benutzeroberfläche auszugeben und den Speicherpfad neu zu setzen. Das Zähler-Objekt wird aufgefordert, alle seine Zähler zurückzusetzen.

### ***private void addBuzzerListener()***

Setzt einen Listener zum Abhören von Tätigkeiten, die an der Buzzerbox statt finden.

### ***private void addViewListener()***

Setzt einen Listener zum Beobachten von Vorgängen an der Benutzeroberfläche der Client-Anwendung (View).

### ***private void actionPerformed(PropertyChangeEvent e)***

Ermittelt, welche Tasten und Knöpfe der Buzzerbox gedrückt wurden und startet die damit jeweils verbundene Aktion. *Für den grünen Knopf ist eine entsprechende Behandlungsroutine hier ebenfalls aufzurufen.*

### ***private void setErrorTCP\_Connection()***

Fehlerbehandlungsroutine für eine nicht vorhandene oder möglicherweise fehlerhafte Verbindung zum Server. Die Fehlermeldung wird daraufhin auf der Benutzeroberfläche angezeigt.

### ***private void setErrorDirNotAcessable()***

Fehlerbehandlungsroutine für den Fall, dass der eingestellte Speicherpfad nicht vorhanden oder mit ungenügenden Rechten ausgestattet ist. Die Meldung wird an der Benutzeroberfläche ausgegeben.

## ***Package model***

Das Package model stellt das Model im Sinne des MVC dar. Hier werden die einzelnen Buzzertool-Objekte definiert, die vom Controller benutzt werden. *Weitere Klassen für den grünen Knopf kommen ebenfalls in dieses Package.*

Klassen:

### **AudioRecorder**

Die Klasse AudioRecorder übernimmt das komplette Handling der Audioaufnahmen.

Weil die Aufzeichnung auch gleichzeitig gespeichert werden soll, wird ein zusätzlicher Thread benötigt, der von einer inneren Klasse Recorder gestartet wird.

*Methoden:*

***public AudioRecorder()***

Legt ein neues Objekt der inneren Klasse Recorder an, um den Thread nutzen zu können.

***public void startRecording()***

Gibt den Aufzeichnungswunsch an das Thread-Objekt weiter.

Diese Methode wird aus dem Controller heraus aufgerufen, wenn der rechte Buzzer gedrückt wird.

***public void stopRecording()***

Teilt dem Thread-Objekt mit, dass die Aufzeichnung beendet werden soll.

Diese Methode wird aus dem Controller heraus aufgerufen, wenn der rechte Buzzer losgelassen wird.

***public void setAudioFormat(int sampFQ, int cha)***

Setzt das Audioformat der Sprachaufzeichnung.

***public void openWinAudioSettings()***

Um weitere Einstellungen im Betriebssystem tätigen zu können, wird über diese Methode das Windows-Programm SNDVOL32.EXE aufgerufen.

***public void setChannels(int channels)***

Setzt den für die Aufnahme zu verwendeten Kanal.

***public void setSampFreq(int sampFreq)***

Setzt die für die Aufnahme benötigte Frequenz.

***public void setPath(String pa)***

Setzt den Ort, an dem die Audiodatei gespeichert werden soll.

***public String getPath()***

Gibt den Pfad zurück, in dem die Audioaufzeichnung gespeichert wird.

Die innere Klasse Recorder für den beteiligten Thread enthält darüber hinaus folgende Methoden:

***public void run()***

Startet den internen Audio-Thread.

***public String getPathWithFileName()***

Ermittelt Speicherpfad und Dateinamen für die abzuspeichernde Audioaufzeichnung.

***private File writeNewFile()***

Speichert die Sprachaufzeichnung auf der Festplatte, sofern der angegebene Pfad gültig ist.

***public void startRecord()***

Steuert das Mikrofon an und startet die Sprachaufzeichnung. Gleichzeitig wird die Aufzeichnung durch Aufruf der Methode `writeNewFile()` abgespeichert. Eventuell auftretende Fehler werden abgefangen.

***public void stopRecord()***

Hier wird die Sprachaufzeichnung gestoppt.

**ScreenShot**

Die Klasse `ScreenShot` übernimmt das Handling der Screenshots. Auch hier wird eine innere Klasse zum Zeichnen des Screenshots in die zu speichernde Datei (dieser Vorgang läuft ebenfalls über einen Thread ab) benutzt.

*Methoden:*

***public BufferedImage getScreenshot()***

Hier wird der Screenshot erzeugt und (sofern keine Fehler auftreten) an den Controller zurückgegeben. Diese Methode wird vom Controller aufgerufen, sobald der linke Buzzer der Buzzerbox gedrückt wird.

***public void makeScreenshot()***

Erzeugt ein Objekt der inneren Klasse und veranlasst dieses, den Screenshot zu zeichnen. Somit wird der interne Thread gestartet.

***public void saveImage(BufferedImage image, String path)***

Über die Hilfsklasse `UseLabUtilities` wird der Dateiname für den Screenshot ermittelt und das Bild im angegebenen Pfad abgespeichert.

***public String getLastPath()***

Gibt den aktuell eingestellten Speicherpfad zurück.

Die innere Klasse `CaptureThread` für den beteiligten Thread enthält darüber hinaus folgende Methoden:

***private void shootMultiScreen()***

Zeichnet das Bild, das vom Server empfangen wurde, neu.

***private BufferedImage addCursorToImage(BufferedImage im)***

Fügt den Mauszeiger im gezeichneten Bild hinzu. Zurückgegeben wird ein gesamter neu gezeichneter Screenshot.

## **QuestionCounter**

Die Klasse QuestionCounter ist zuständig für den internen Aufgabenzähler und die Zusammenstellung der Informationen zum durchgeführten Test.

*Methoden:*

### **public void setPath(String path)**

Setzt den Pfad für die Ausgabedatei (Informationstextdatei).

### **private void setStartTime()**

Legt über die Hilfsklasse UseLabUtilities die Anfangszeit der gerade bearbeiteten Aufgabe fest.

### **private void queryCounter()**

Falls die Aufgabe nicht erfolgreich gelöst wurde, wird dies im String über setQuestionInfo() vermerkt. Anschliessend wird der String in einer ArrayList für die spätere Dateiausgabe zwischengespeichert. Diese Methode ist vor allem dazu da, um abzufragen, wann bei Start einer neuen Aufgabe der weisse Knopf nicht gedrückt wurde.

### **private void setQuestionInfo()**

Ermittelt die Bearbeitungszeit für die gerade abgelaufene Aufgabe und komplettiert den String um die restlichen Informationen. Der Informationsstring ist somit folgendermassen aufgebaut:  
Aufgabennummer → benötigte Zeit in Minuten:Sekunden → Erfolgreich, Ja/ Nein

### **private void saveInfo()**

Schreibt alle Informationen aus der ArrayList (Container für die Informationsstring – 1 Informationsstring pro Aufgabe) in eine Textdatei raus.

### **public int getSuccess()**

Liefert die Anzahl der richtig bearbeiteten Aufgaben im aktuellen Test.

### **public void handleSuccessMessage()**

Erhöht die Anzahl der erfolgreich gelösten Aufgaben um 1 Einheit und lässt den Informationsstring über setQuestionInfo() mit den noch fehlenden Angaben komplettieren.

### **public void reset()**

Zunächst wird hier überprüft, ob sich noch Informationen bzgl. der letzten Aufgabe im Informationsstring befinden (besonders wichtig, wenn diese nicht erfolgreich gelöst werden konnte).

Anschliessend alle internen Zähler für den nächsten Test auf den Anfangswert zurücksetzen. Der Inhalt der ArrayList (alle Informationsstrings für die bearbeiteten Aufgaben) wird über saveInfo() in eine Textdatei ausgelesen.

### **public int getTestNr()**

Gibt die laufende Nummer des aktuellen Tests zurück (wichtig für die Erstellung des Unterordners im aktuellen Speicherpfad).

***public boolean getSuccessFlag()***

Gibt an, ob die letzte Aufgabe korrekt bearbeitet worden ist oder nicht.

***public int getQuestionCounter()***

Liefert die Anzahl aller bisherigen Aufgaben im aktuellen Test.

***public void incrementQuestion()***

Erhöht den Aufgabenzähler um 1, setzt den Informationsstring neu für die nächste Aufgabe und ermittelt über queryCounter() die Anfangszeit der Bearbeitung der kommenden Aufgabe. Ausserdem wird noch nach der letzten Aufgabe abgefragt, damit im Falle einer nicht korrekt gelösten Aufgabe (weisser Knopf wird dann nicht gedrückt) auch diese Information über den Fehlschlag gespeichert wird.

***Package observer***

Im Sinne des MVC dient das Package observer zum "Beobachten" der Buzzerbox und teilt Veränderungen am Zustand der Buzzerbox (Knöpfe/Buzzer gedrückt oder nicht gedrückt) der Client-Anwendung mit.

Klassen:

***PollObserver***

Die Klasse PollObserver ist die Schnittstelle zur Buzzerbox, im Sinne des MVC. Sie beinhaltet eine innere Klasse PollDaemon, die threadbasiert die Vorgänge an der Buzzerbox laufend abfragt.

Methoden:

***public PollObserver()***

Initialisiert ein PropertyChangeSupport-Objekt, welches dem Controller über Vorgänge an der Buzzerbox unterrichten soll.

***private void initController()***

Initialisiert die Buzzerbox und den Controller der Buzzerbox.

***private PollDaemon()***

Startet den Thread für Überwachung der Buzzerbox.

***public void run()***

Prüft mittels Polling nach, ob einer der grossen Buzzer oder der kleinen Schalter an der Buzzerbox gedrückt wurde und leitet entsprechende Handlingmethoden ein, d.h. über das PropertyChangeSupport-Objekt wird dem Controller mitgeteilt, dass ein Ereignis an der Buzzerbox stattgefunden hat. *Die Abfrage nach dem bisher unbelegten grünen Knopf ist bereits eingebaut und braucht nicht mehr implementiert werden.*

## ***Package resources***

Das Package resources beinhaltet zwei Hilfsklassen, die vom Programm intern benötigt werden, und auf die alle anderen Klassen Zugriff haben.

Klassen:

### **Resources**

Die Klasse Resources beinhaltet lediglich wichtige Pfadangaben zu Ressourcen sowie einige Konstanten, die überall im Programm benötigt werden. Somit sind hier keine eigenen Methoden nötig.

### **UseLabUtilities**

Die Klasse UseLabUtilities stellt ein einheitliches Datums- und Zeitformat zur Verfügung und beinhaltet einige Hilfsmethoden zur Ermittlung der aktuellen Zeit.

*Methoden:*

#### ***public static int getHour()***

Liefert die aktuelle Stunde.

#### ***public static int getMinute()***

Liefert die aktuelle Minute.

#### ***public static int getSecond()***

Liefert die gerade angebrochene Sekunde.

#### ***public static String getTime()***

Liefert die Zeitangabe in einem String der Form "hh-mm-ss"

#### ***public static String getTimeMil()***

Liefert die Zeitangabe in einem String der Form "hh-mm-ss-msms".

Aus diesem String wird der Dateiname der abzuspeichernden Sprachaufzeichnungen und Screenshots abgeleitet.

## ***Package view***

Das Package view stellt den View im Sinne des MVC dar.

Hier wird die gesamte Benutzeroberfläche des Clients (die GUI) definiert, und Eingaben sowie die anderen Aktionen auf der Oberfläche verwaltet. Entsprechende Änderungsmitteilungen werden an den Controller weitergeleitet bzw. Ausgaben, die von Aktionen an der Buzzerbox abhängen, vom Controller empfangen und umgesetzt.

Klassen:

### **CaptureView**

Die Klasse CaptureView stellt die eigentliche View im Sinne von MVC dar. Hier befindet sich die gesamte GUI der Client-Anwendung.

*Methoden:*

***private void init()***

Initialisiert die GUI mitsamt den Tabs und Buttons, legt allgemeine Position und Grösse der Benutzerschnittstelle fest, fügt Listener hinzu und nimmt allgemein gültige Einstellungen für die GUI vor.

***private void addListener()***

Verbindet die Buttons mit entsprechenden Listnern, die darauf horchen, wenn die Buttons mit der Maus aktiviert werden, und die damit verbundenen Aktionen/Methodenaufrufe auslösen.

***private void exitView()***

Blendet per Popup eine Sicherheitsabfrage ein, ob die GUI tatsächlich verlassen werden soll, und beendet im Falle eines Klicks auf den Button JA das Programm.

***private JPanel getTab1Content()***

Initialisiert die Inhalte des Tabs "Info" bzgl. Position, Grösse, Aussehen (hier wird auch die Grösse des abgebildeten Screenshots festgesetzt) und stattet den dort vorhandenen Button "Open" (Öffnen und Anzeigen des Speicherpfads) mit einem Listener aus.

***private JPanel getTab2Content()***

Verbindet die Panels für "Audio", "SavePath" und "CaptureServer" (siehe unten) zu einem Tab "Config" und legt das Layout des Tabs fest.

***private JPanel getAbout()***

Initialisiert die Inhalte des Tabs "About" bzgl. Position, Grösse, Aussehen.

***private void getPath()***

Methode zum Behandeln des Popup-Menüs, das aufgerufen wird, wenn man den Button rechts neben dem Textfeld "Save Path" auswählt – hier wird der Speicherpfad ausgewählt und gesetzt.

***private void makeDirectory()***

Legt im Speicherpfad einen Unterordner für die Dateien des aktuellen Tests an.

***public void setPath(String path)***

Setzt den Speicherpfad, in dem Textdateien, Audiodateien und Screenshots gespeichert werden sollen.

***private JPanel getPathPanel()***

Initialisiert die Inhalte der Fläche "SavePath" im Tab "Config" bzgl. Position, Grösse und Aussehen.

***private JPanel getAudioPanel()***

Initialisiert die Inhalte der Fläche "Audio" im Tab "Config" bzgl. Position, Grösse, Aussehen und stattet den dort vorhandenen Button mit einem Listener aus.

***private JPanel getNetworkPanel()***

Initialisiert die Inhalte der Fläche "CaptureServer" im Tab "Config" bzgl. Position, Grösse, Aussehen und stattet den dort vorhandenen Button "Apply" (mit Server verbinden – in dem Fall werden die angegebene IP-Adresse und Portnummer auf Korrektheit überprüft) mit einem Listener aus.

***public void setConnetionOk(boolean ok)***

Schaltet die Anzeige von rot (disconnected) auf grün (connected), abhängig davon, ob die Verbindung zum Server zustande gekommen ist. Dies wird über den Parameter "ok" entschieden.

***public static void main(String[] args)***

Startet die Benutzeroberfläche.

***public void actionPerformed(ActionEvent e)***

Wird beim Betätigen des Buttons neben der Anzeige des aktuell eingestellten Speicherpfads aufgerufen und veranlasst, dass der Inhalt des Pfads angezeigt wird. Default-Einstellung ist "C:\".

***public String getSavePath()***

Gibt den aktuellen Speicherpfad zurück.

***public void setOutText(String t)***

Zeigt Textausgaben im Textfeld "Actions" an.

***public void setResizedImagePath(String resizedImagePath)***

Setzt den Pfad für den skalierten Screenshot.

***public void refreshView(String path)***

Wenn ein Screenshot gemacht wurde, so wird in dieser Methode die Skalierung des übertragenen Bildes veranlasst, damit der Screenshot auf der Benutzeroberfläche angezeigt werden kann.

***public int getSelectedSampFreq()***

Zeigt die ausgewählte Frequenz im entsprechenden Textfeld der GUI an.

***public int getSelectedChannel()***

Zeigt den ausgewählten Kanal im entsprechenden Textfeld der GUI an.

***public void openWinAudioSettings()***

Startet das Windows-Programm SNDVOL32.EXE zum Vornehmen von Einstellungen für die Sprachaufzeichnung.

***public void openSavePath()***

Öffnet den Speicherpfad direkt im Windows-Explorer.

***public void switchRecordIcon()***

Schaltet zwischen den Bitmaps "Tonaufnahme läuft" und "keine Tonaufnahme"

um.

***public void getIPAddress()***

Prüft eingegebene IP-Adresse auf Gültigkeit und versucht, die Adresse über das Netzwerk zu erreichen.

***public void getPort()***

Nimmt die eingegebene Portnummer entgegen und überprüft sie auf Gültigkeit.

***private String getStringSave(String text)***

Prüft, ob eine Texteingabe des Benutzers leer ist, und wenn ja, setzt diesen String auf "0".

**ImageResizer**

Die Klasse ImageResizer ist eine Hilfsklasse. Sie skaliert den letzten Screenshot neu, damit dieser auf der Benutzeroberfläche angezeigt werden kann.

*Methoden:*

***public void paint(Graphics g)***

Zeichnet die Grafik.

***public void resize(String path)***

Liest das Bildobjekt aus dem gegebenen Pfad heraus, skaliert es auf einen konstanten Default-Wert und zeichnet es neu.

***public void resize(String path, int x, int y)***

Liest das Bildobjekt aus dem gegebenen Pfad heraus, skaliert es in der angegebenen x- und y-Richtung und zeichnet es neu. Das Bild kann damit vergrößert oder verkleinert werden.

**JTextFieldFilter**

Die Klasse JTextFieldFilter ist eine Hilfsklasse und überprüft die Gültigkeit einer IP-Adresse.

*Methoden:*

***public JTextFieldFilter(String acceptedchars)***

Konstruktor. Hier wird festgelegt, welche Zeichen eine IP-Adresse beinhalten darf.

***public void insertString (int offset, String str, javax.swing.text.AttributeSet attr)***

Hier wird getestet, ob der übergebene String den Einstellungen bzgl. der Gültigkeit einer IP-Adresse entspricht oder nicht.

## B2 VirtualDub

Virtual Dub liegt derzeit in den Versionen 1.7.5 sowie 1.7.7 als Verknüpfung auf dem Desktop des Schnittrechners bereit. Welche der beiden Versionen letztendlich benutzt wird, spielt für die Zusammenstellung der Filme keine Rolle.

Für den Fall, dass das Schneiden der Filme **nicht** automatisch über das Operatingtool erfolgen soll (sondern eigenständig), sind die nachfolgenden Schritte auszuführen.

### Erstellung des Endfilms über ein AviSynth-Skript

1. VirtualDub über die Desktopverknüpfung öffnen.
2. Den Ordner mit den fertigen AviSynth-Skripten öffnen  
(C:\UselabOrdner\AviSynthSkripte, siehe Abbildung B2 – 1)

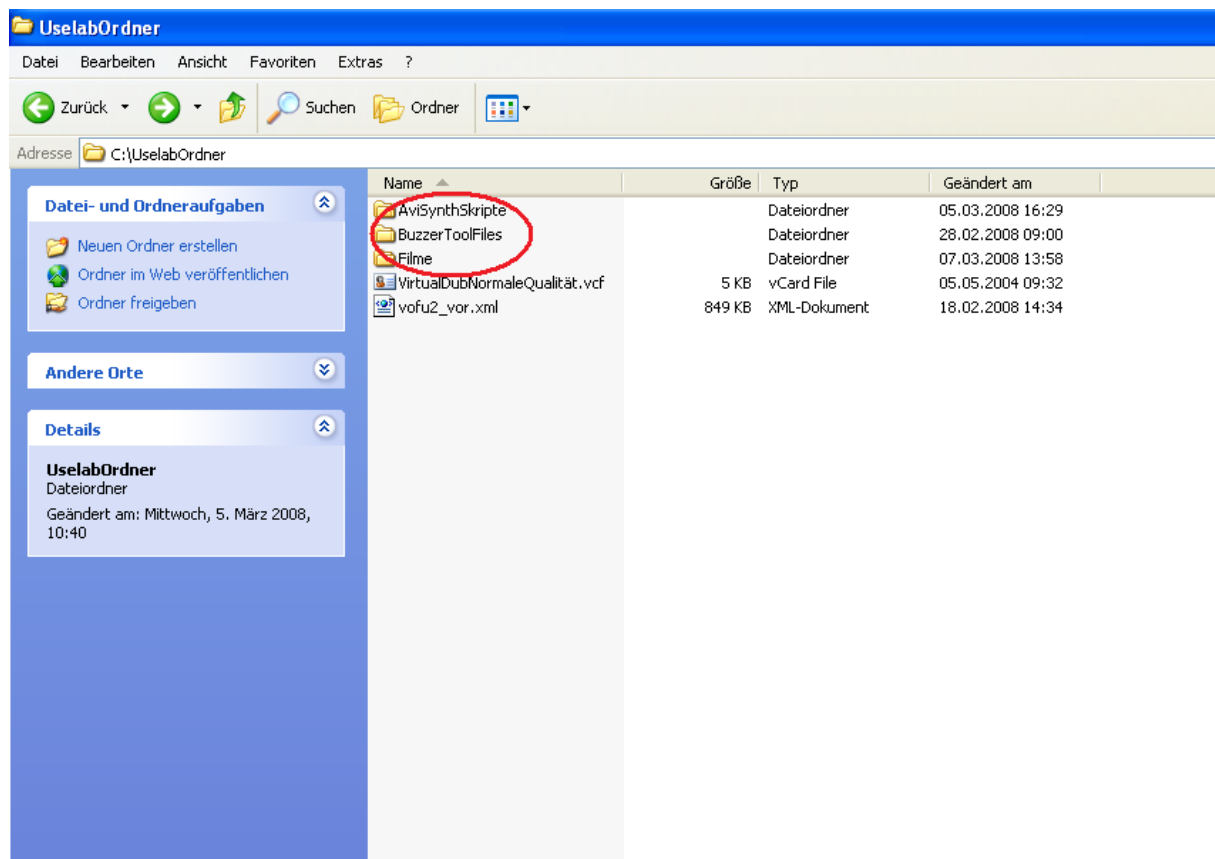


Abb. B2- 1: Der Uselabordner enthält ein Verzeichnis mit den AviSynth-Skripten

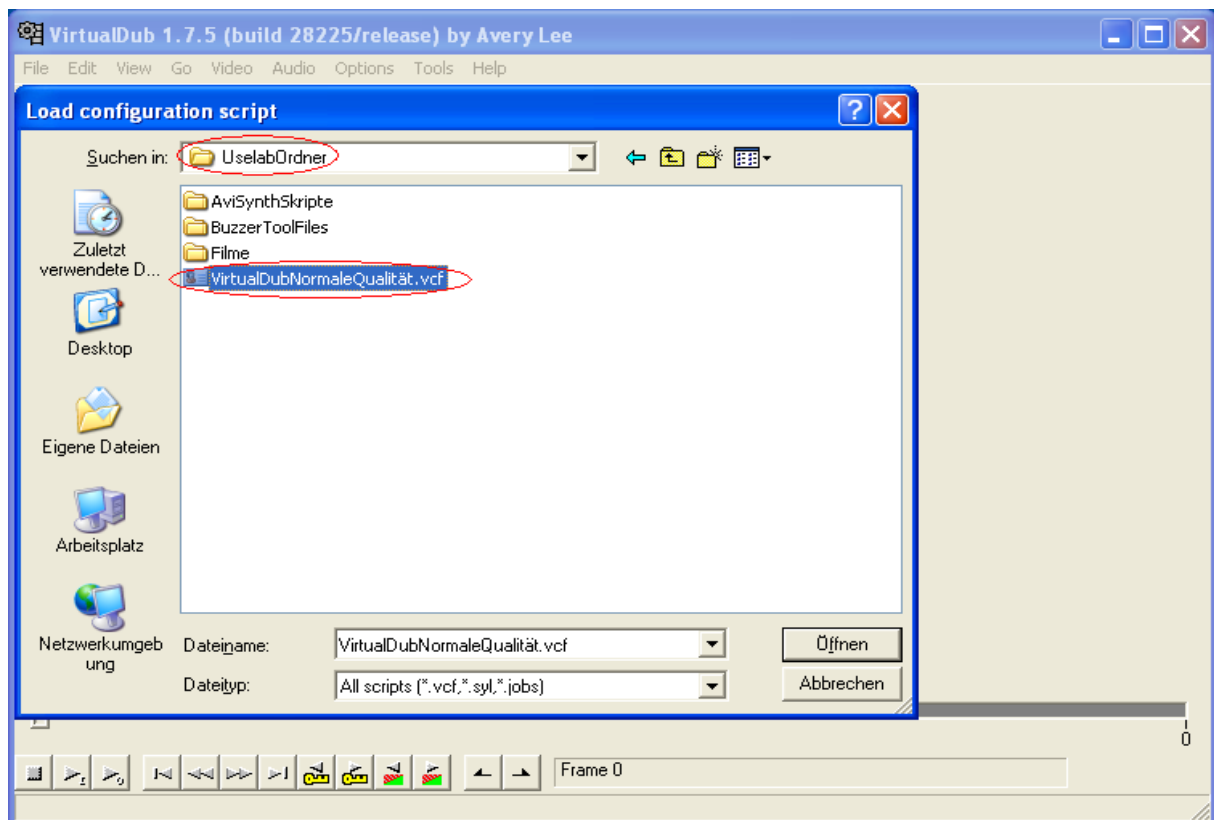


Abb. B2- 2: Karte laden

3. CTRL+L drücken und die Karte „VirtualDubNormaleQualität“ laden (diese VCF-Datei befindet sich im Uselabordner, siehe obige Abbildung B2 – 2)
4. Erstes AviSynth-Skript nach VirtualDub hineinziehen, wie in Abbildung B2 – 3 gezeigt.

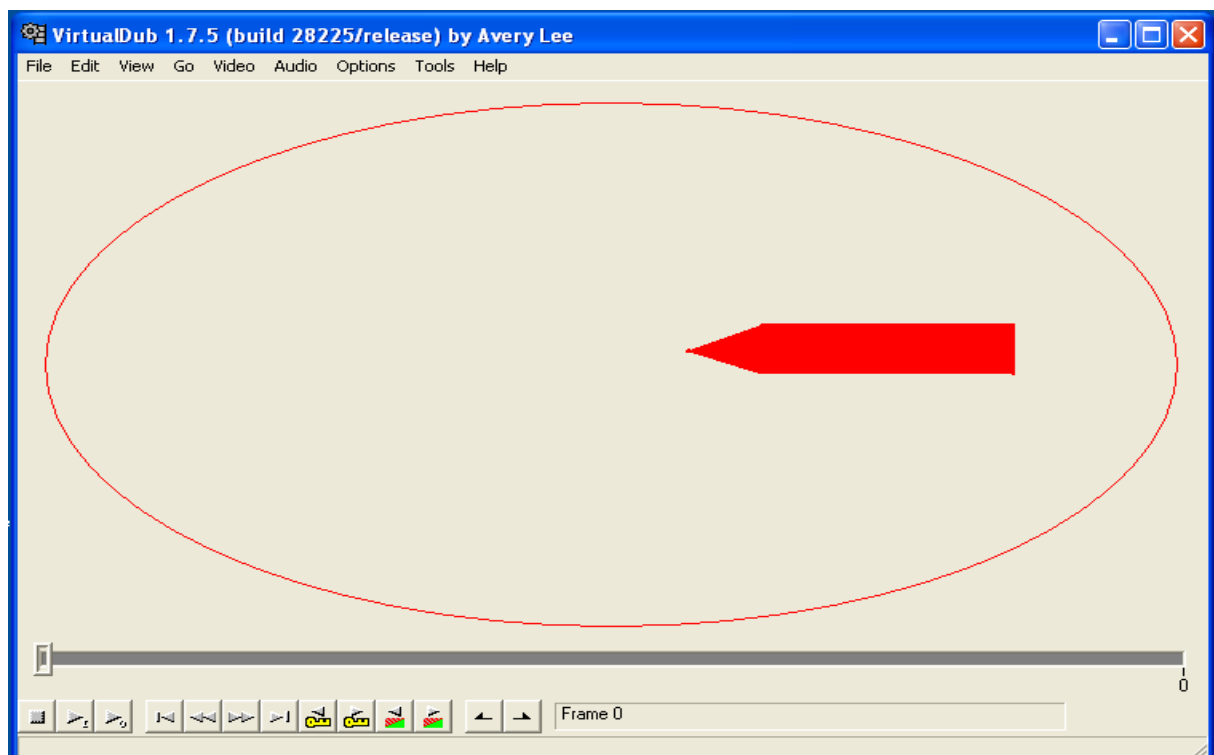


Abb. B2- 3: AviSynth-Skript in den Hauptbereich ziehen

5. Taste F7 drücken und Zielpfad des fertigen Films bestimmen. Fertige Filme sollten in das Verzeichnis C:\UselabOrdner\Filme abgelegt werden. Dem fertigen Film einen eindeutigen Namen (zum Beispiel „Eyetracker\_Test\_1.avi“) vergeben. Haken bei „Don't run job now“ setzen. Siehe hierzu Abbildung B2 – 4.

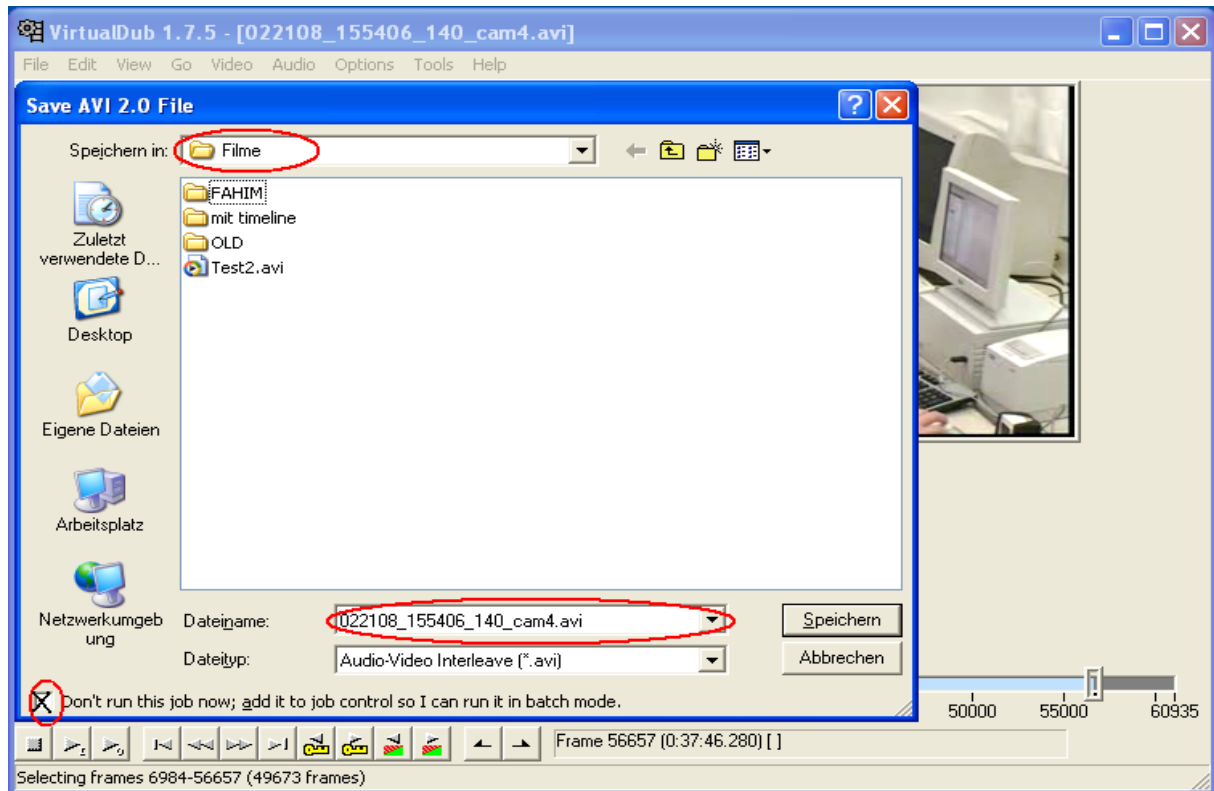


Abb. B2- 4: Einstellungen vor dem Rendern

6. Schritte 4 und 5 für weitere Skripte wiederholen (falls vorhanden).
7. Taste F4 und dann auf „Start“ drücken, um die Jobqueue abzuarbeiten (siehe Abbildung B2 – 5). Die Filme werden nacheinander automatisch fertiggestellt.

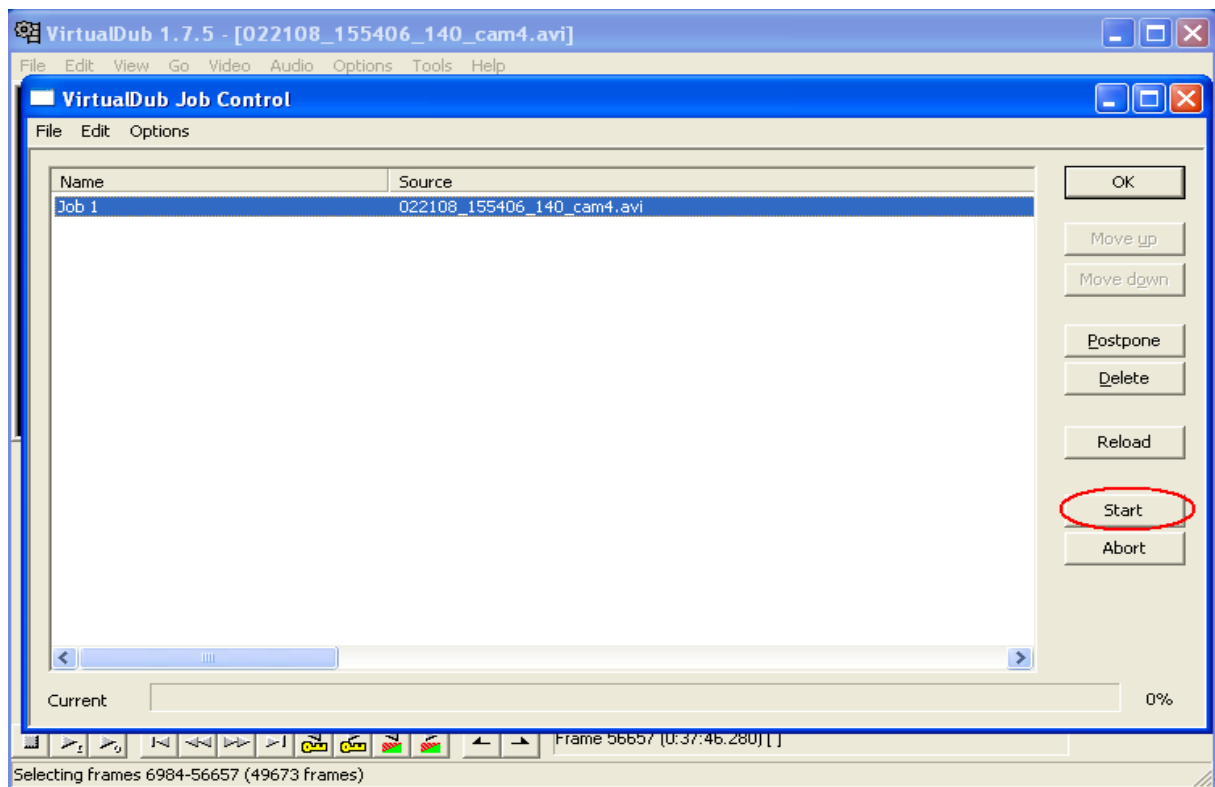


Abb. B2- 5: Job Control von VirtualDub

8. Zum Abschluss Queue leeren, indem alle Jobs aus der Liste entfernt werden.

## Manuelles Synchronisieren der Einzelfilme durch Zurechtschneiden derselben

Da es unter Umständen sein kann, dass die Filmaufnahmen der einzelnen Kameras nicht vollständig synchron zueinander sind, kann VirtualDub dazu verwendet werden, eine Synchronisierung der Einzelfilme manuell durchzuführen.

1. VirtualDub über die Desktopverknüpfung öffnen.
2. Einen Rohfilm (**muss** im AVI-Format vorliegen) nach VirtualDub hineinziehen. Es ist günstig, dass auf diesem Rohfilm auch der Bildschirm des Testrechners zu sehen ist. Daher bieten sich entweder Kamera 3 oder Kamera 4 für diesen Schritt an.
3. Einen markanten Punkt (z.B. eine bestimmte Bewegung der Testperson) am Anfang des Films suchen. Hinweis: Mit dem Schieberegler kann man den Film vor – und zurückspulen lassen.

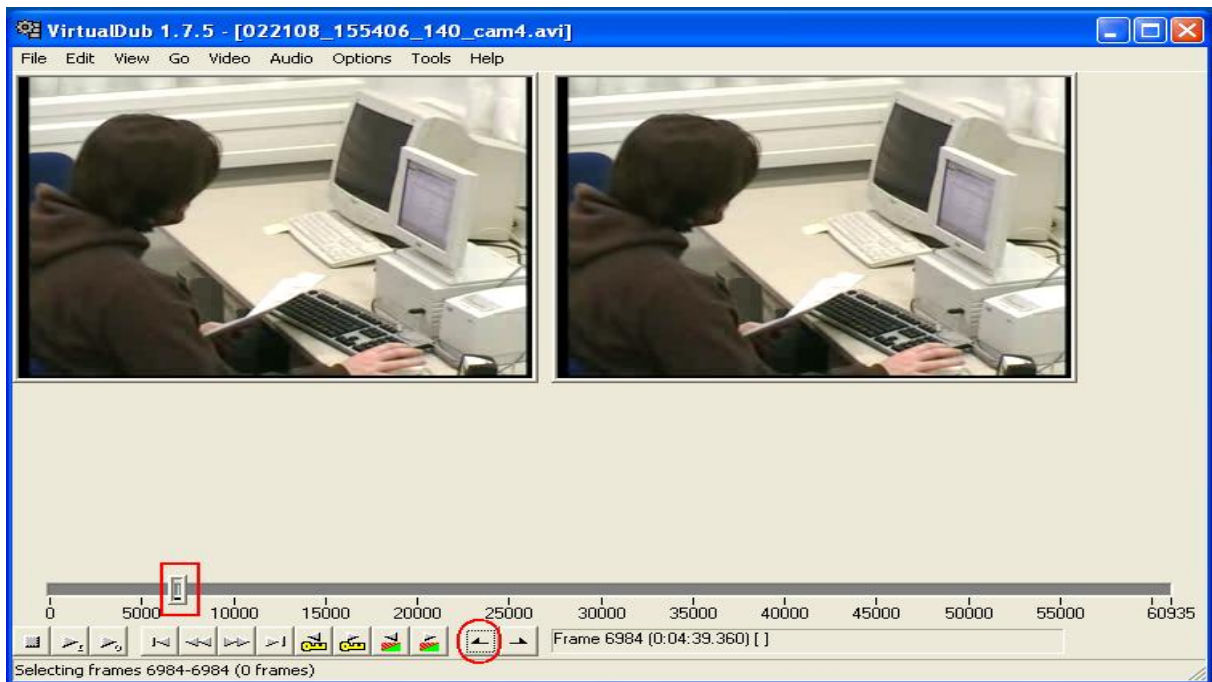


Abb. B2- 6: Anfangspunkt setzen

4. Button zum Setzen der Startposition betätigen (Abbildung B2 – 6).
5. Schritt 3 für das Ende des Films wiederholen.
6. Button zum Setzen der Endposition betätigen (Abbildung B2 – 7).

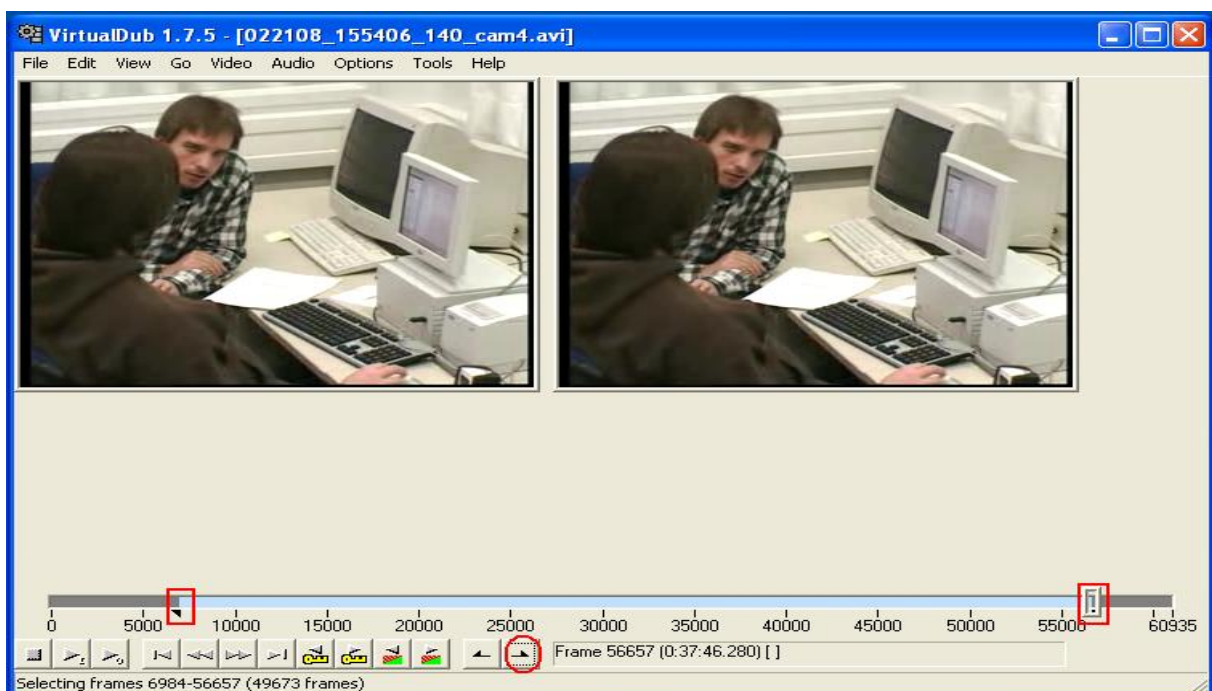


Abb. B2- 7: Endpunkt setzen

7. Taste F7 drücken und Zielpfad des geschnittenen Films bestimmen. Dieser Film sollte sich am Ende im gleichen Verzeichnis wie der originale Rohfilm befinden.
8. Schneidevorgang starten.
9. Schritte 3-8 für alle anderen Kameras wiederholen. Man sollte sich hierbei an den markanten Punkten des zuerst geschnittenen Films orientieren.

## B3 Operatingtool

Das Operatingtool stellt eine Arbeitserleichterung des Operators dar. Mit ihm werden nahezu alle Arbeitsschritte des Operators automatisiert. Die einzelnen Workflows lassen sich sowohl einzeln als auch als Gesamteinheit in einem einzigen Schritt ausführen.

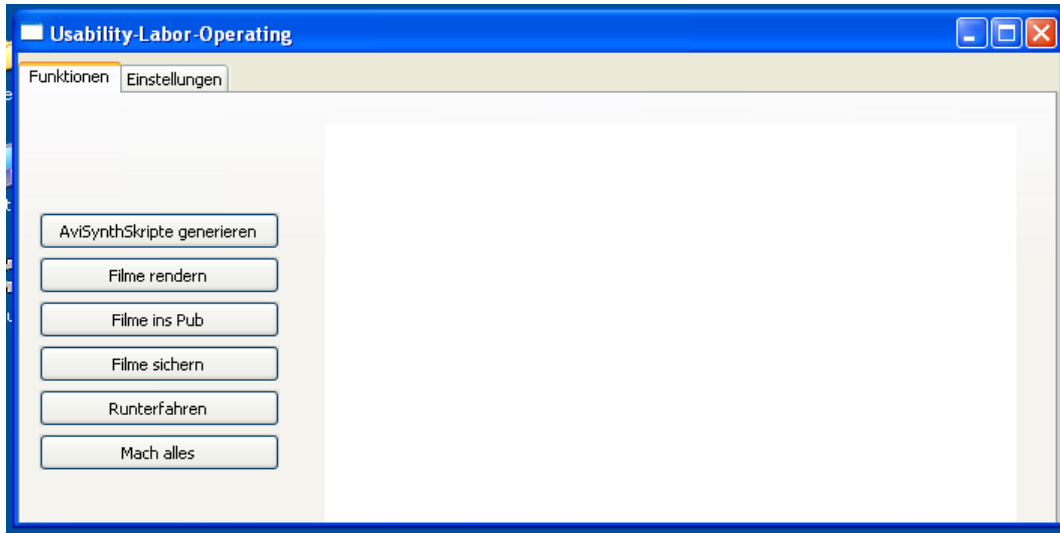


*Abb. B3- 1: Funktionsübersicht des Operatingtools*

Wie der Abbildung B3 – 1 entnommen werden kann, können folgende Schritte durchgeführt werden:

- Unter „AviSynthSkripte generieren“ können für alle (!) vorhandenen Rohfilme die entsprechenden AviSynth-Skripte automatisch erstellt werden. Zu beachten ist hierbei, dass gleich für jeden aufgenommenen Usability-Test das Skript erstellt wird, nicht nur für einen einzelnen Test. Nach Fertigstellung der Skripte wird eine entsprechende Meldung im Textfenster ausgegeben (siehe Abbildung B3 – 2).

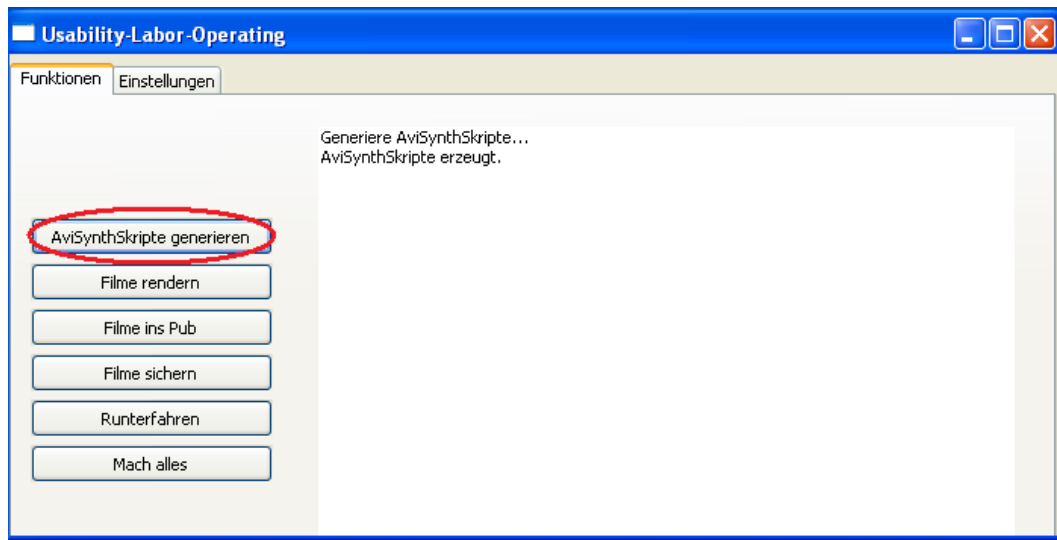


Abb. B3- 2: AviSynth-Skripte erzeugen

- Unter „Filme rendern“ können die erzeugten AviSynth-Skripte in Endfilme umgewandelt werden. Eine Bedienung des Programms VirtualDub ist nicht mehr nötig, dies geschieht vollständig automatisch (siehe Abbildung B3 – 3).

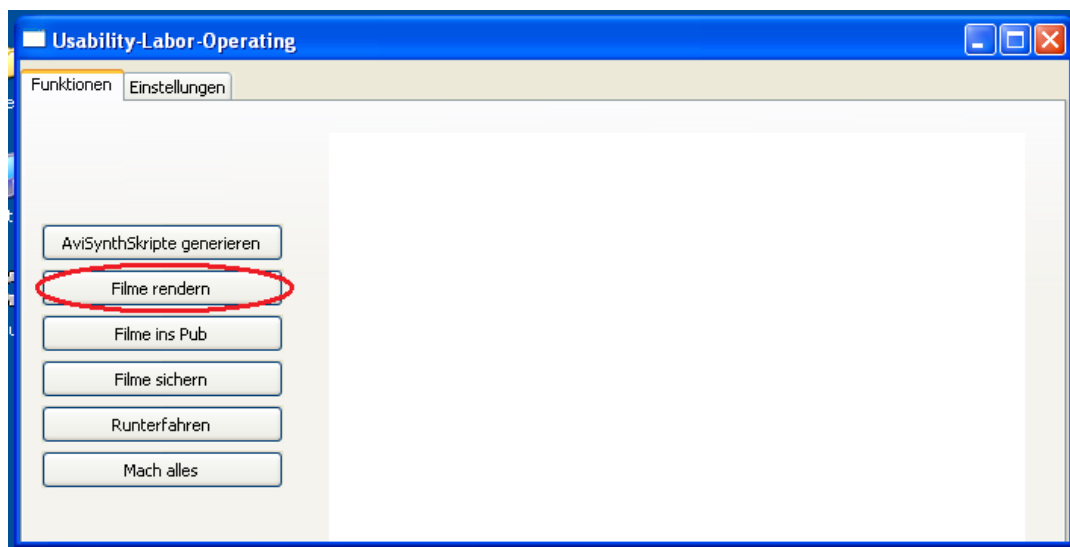


Abb. B3- 3: Filme automatisch über VirtualDub rendern lassen

- Unter „Filme ins Pub“ werden Kopien aller Filme aus dem Ordner C:\Uselabordner\Filme in das öffentliche Verzeichnis des Usability-Labors gestellt. Eine entsprechende Mitteilung erscheint im Textfenster (siehe Abbildung B3 – 4).

**Hinweis:** Momentan wird nur die Mitteilung ausgegeben, hier passiert also nicht viel.

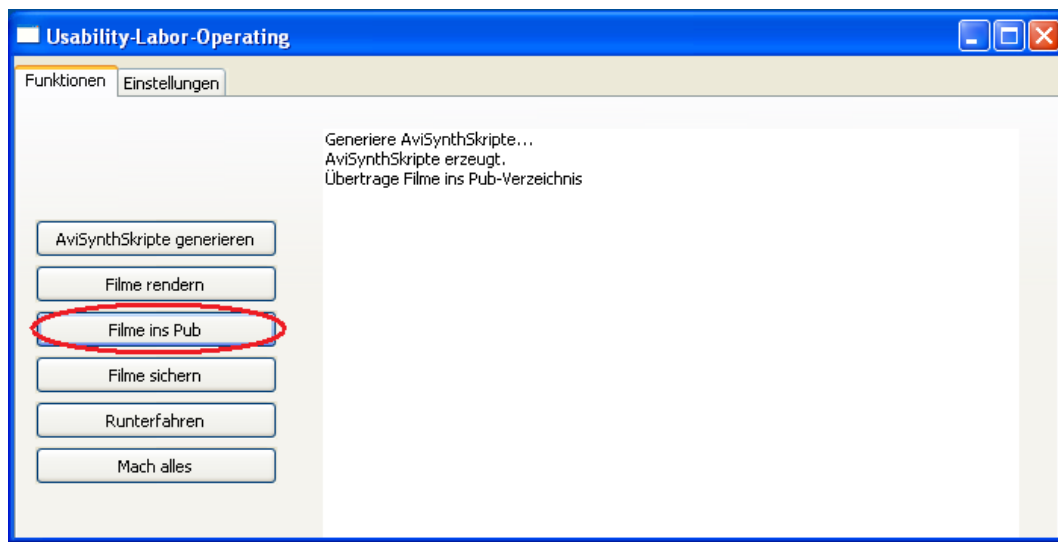


Abb. B3- 4: Filme in das öffentliche Verzeichnis transferieren

- „Filme sichern“: Entspricht dem Arbeitsschritt für die Archivierung von fertigen Filmen. Die Filme werden auf der Linkstation gesichert. Als Meldungen werden sowohl Pfad der Originaldatei sowie Pfad zur Linkstation für jeden Film ausgegeben (siehe Abbildung B3 – 5).

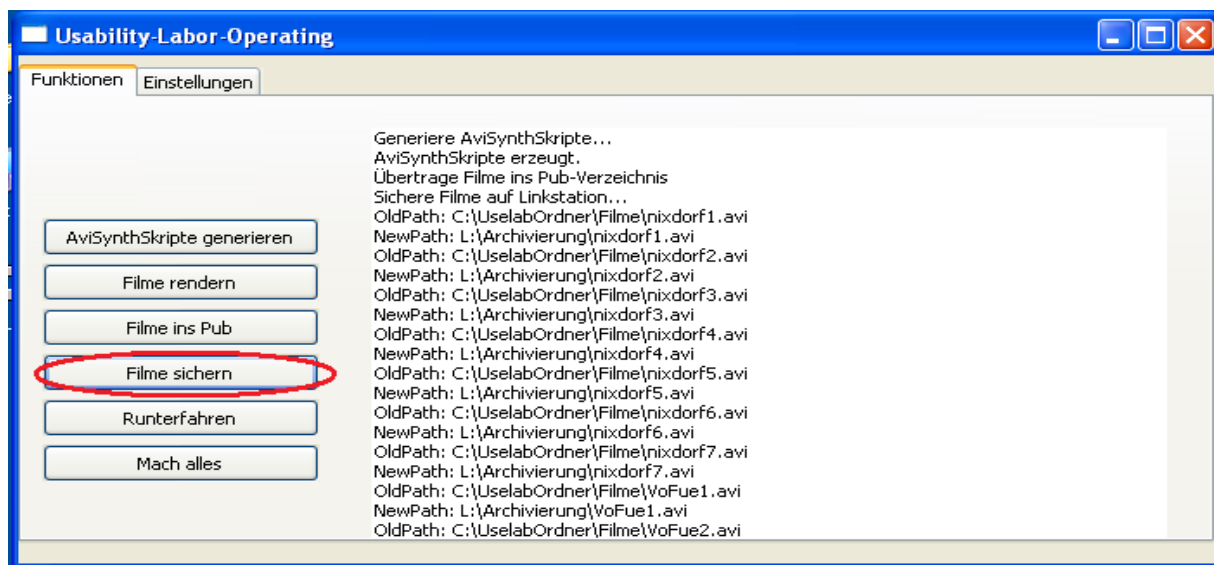


Abb. B3- 5: Filme auf der Linkstation sichern

- „Rechner runterfahren“: Führt den Schnittrechner nach einer Wartezeit von 30 Sekunden runter (siehe Abbildung B3 – 6).  
**Achtung:** Es sollten alle noch laufenden Programme geschlossen werden oder die Beendigung der Programme abgewartet werden !!  
 Alle nicht gespeicherten Daten gehen sonst verloren und noch laufende Anwendungen werden automatisch abgebrochen.  
 Dies ist daher unbedingt nachzuprüfen.

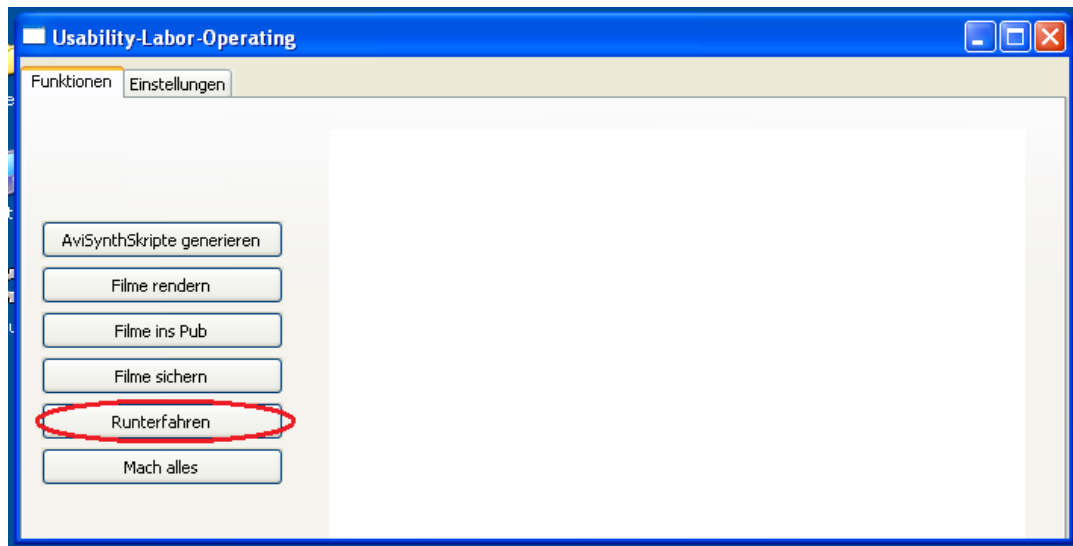


Abb. B3- 6: Rechner runterfahren

- „Mach alles“ führt nacheinander die einzelnen Schritte von der Generierung der Skripte bis einschliesslich zum Runterfahren des Rechners automatisch durch (siehe Abbildung B3 – 7).

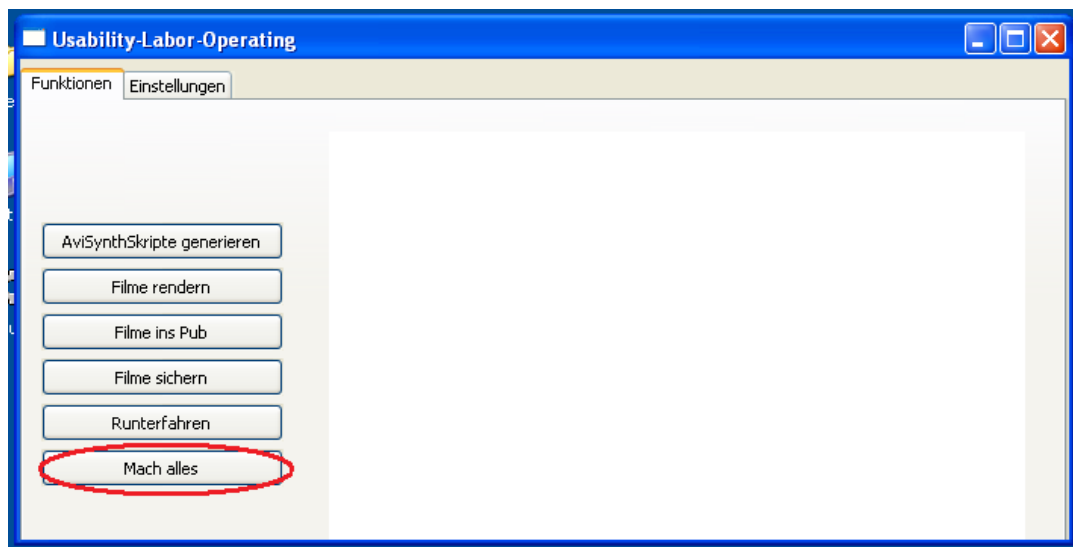


Abb. B3- 7: Komplettdurchlauf

## B4 Acronis

### 1. Acronis True Image Workstation starten

Wahlweise wird Acronis über das Startmenü oder über die Desktop-Verknüpfung gestartet, siehe Abbildung B4 – 1.

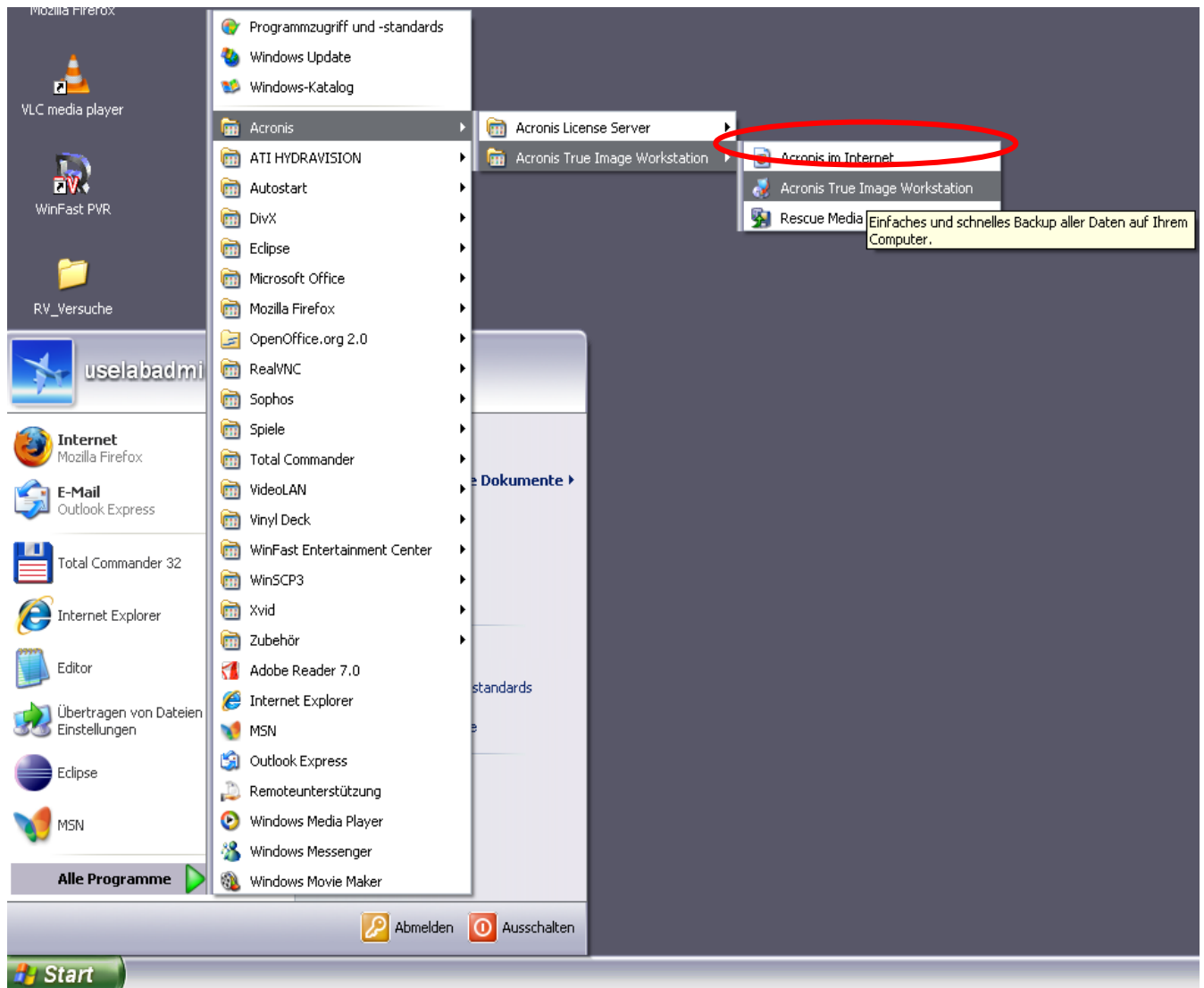


Abb. B4- 1: Acronis über Startleiste

## 2. Daten sichern

### 1. Backup auswählen (Abbildung B4 – 2)

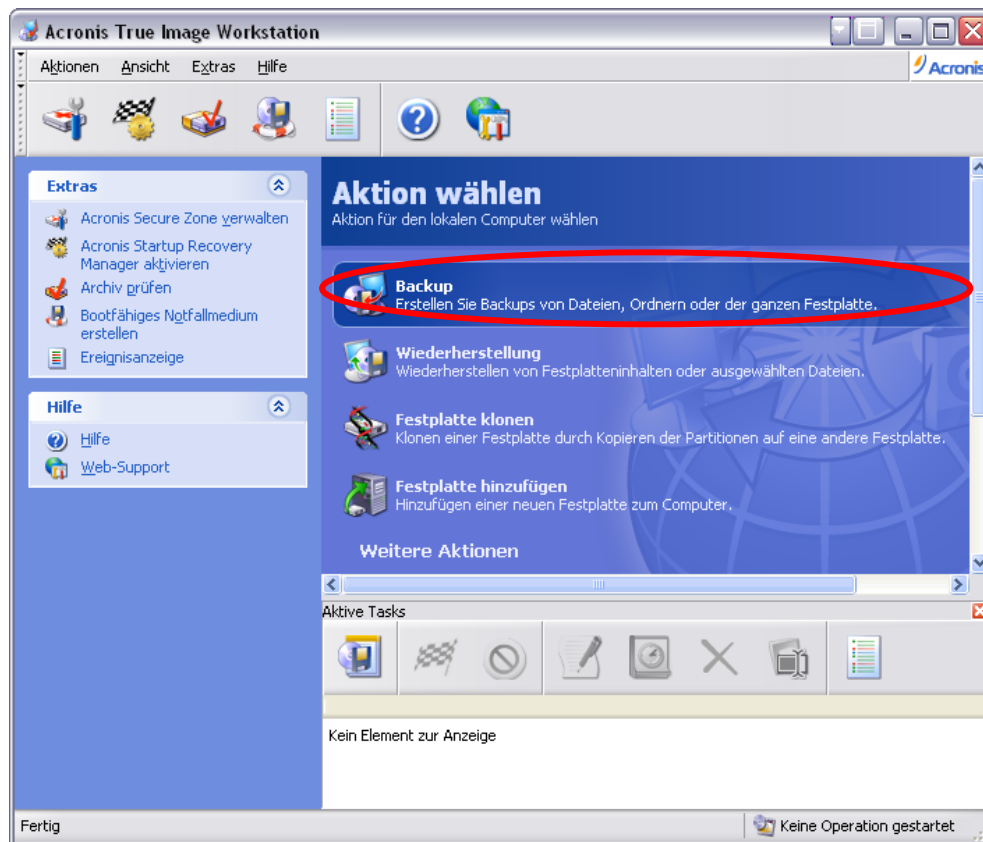


Abb. B4- 2: Auswahl des Backups

### 2. Den nächsten Dialogschritt mit „weiter“ bestätigen (Abbildung B4 – 3)



Abb. B4- 3: Begrüßungsbildschirm des Assistenten

### 3. Typ des Backups auswählen (Abbildung B4 – 4)

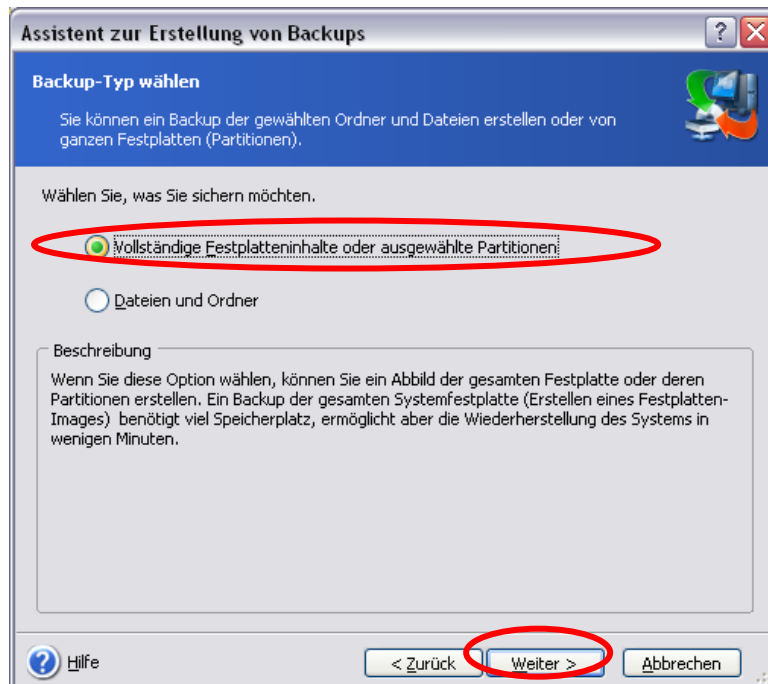


Abb. B4- 4: Art des Backups

- Backup des vollständigen Festplatteninhaltes bei Durchführung einer Komplettsicherung aller Festplatten
  - Backup von Dateien und Ordnern für die Zweitsicherung der Endfilme
- Anschliessend mit „weiter“ fortfahren.

### 4. Auswahl der Partition (Abbildung B4 – 5)

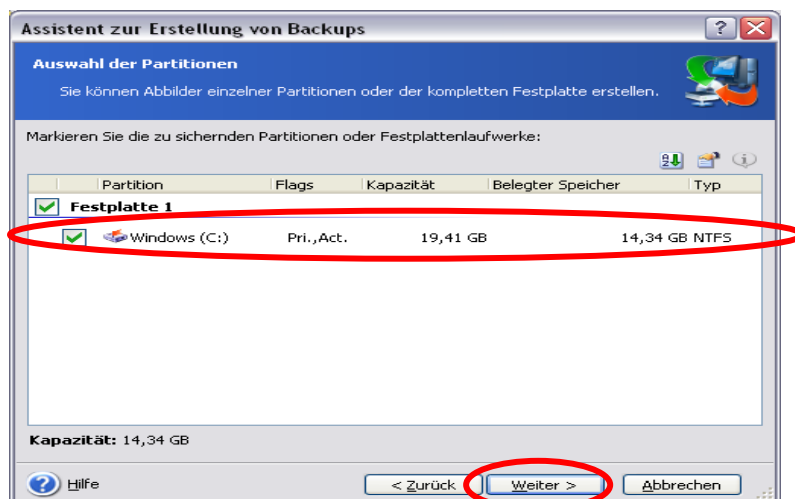


Abb. B4- 5: Quellverzeichnis auswählen

- Auf dem Testrechner sollte stets die C: - Partition gesichert werden
- Auf dem Schnittrechner sind die Partitionen C: und D: zu sichern
- Ebenfalls sollte das G: - Laufwerk der externen Festplatte gesichert werden
- Für den Filmeordner sollte das Verzeichnis L:\Archivierung ausgewählt werden (zur Erinnerung: dieses Verzeichnis befindet sich auf der Linkstation)

### 5. Zielverzeichnis auswählen (Abbildung B4 – 6)

Alle Backups sollten auf der Linkstation im Ordner L:\Backups gespeichert werden. Eine Ausnahme stellt das Backup des Filmeordners dar, da das Ursprungsverzeichnis selbst auf der Linkstation angegeben ist. Hier bietet sich beispielsweise die externe Festplatte G: zur Aufnahme der Backupdatei an.

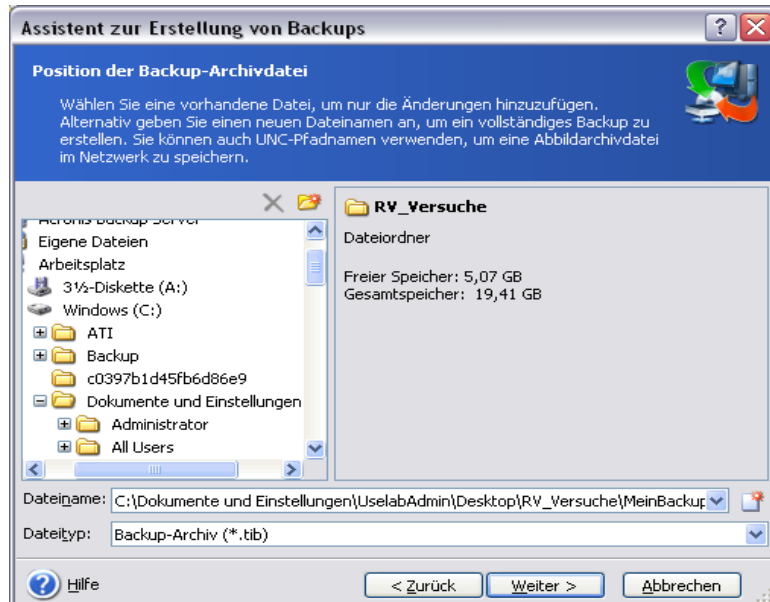


Abb. B4- 6: Zielpfad angeben

## 6. Backup-Modus wählen:

Die Option „Erstelle ein neues vollständiges Backup-Archiv“ erstellt ein exaktes Abbild der gesamten Festplatte. Diese Option ist bevorzugt auszuwählen (vergleiche Abbildung B4 – 7).

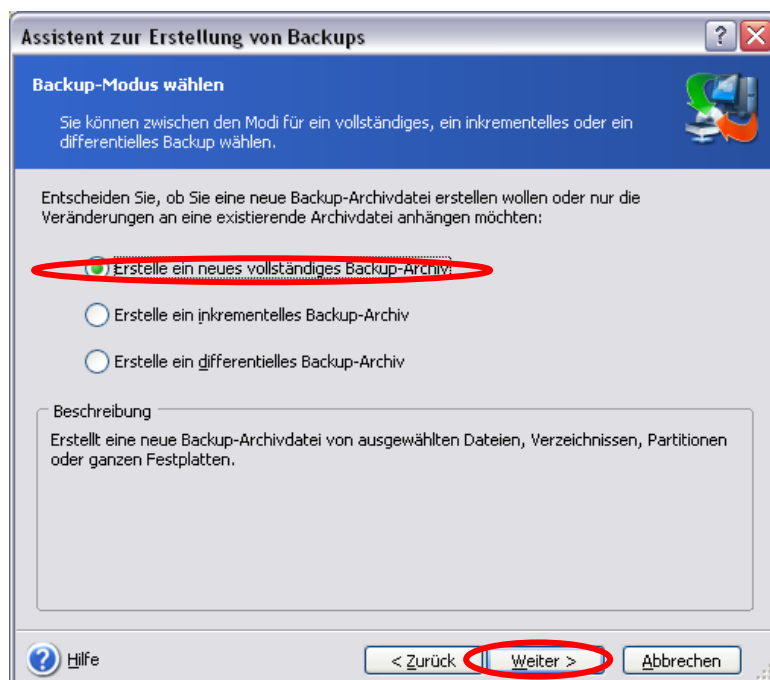


Abb. B4- 7: Auswahl des Archivtyps

## 7. Backup-Optionen wählen

Standardoptionen verwenden (Abbildung B4 – 8). Bei der Installation wurden diese manuell so abgeändert, dass grundsätzlich die maximal mögliche Kompression auf die Daten angewandt wird.

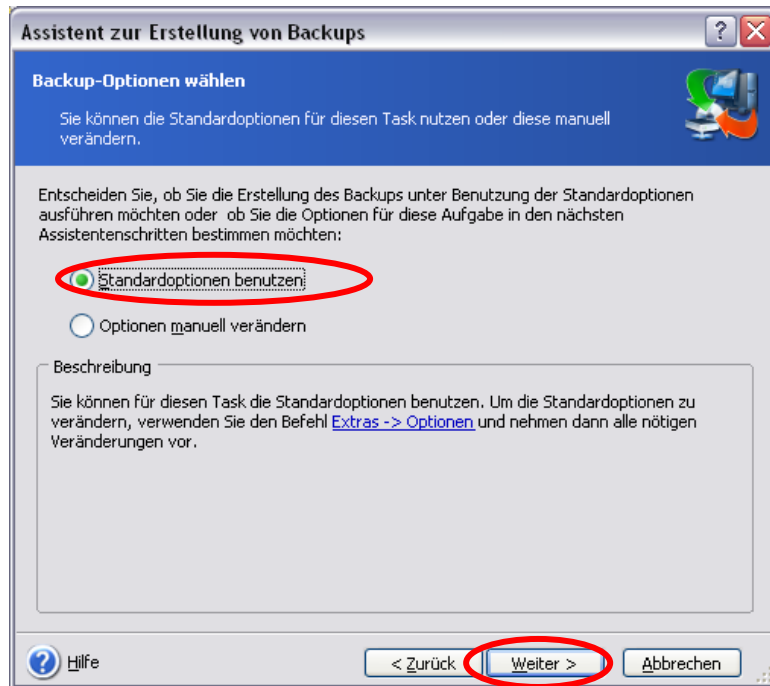


Abb. B4- 8: Auswahl der Optionen

## 8. Archiv Kommentieren (Abbildung B4 – 9)

Hier sollte eine aussagekräftige Beschreibung des Backups angegeben werden.

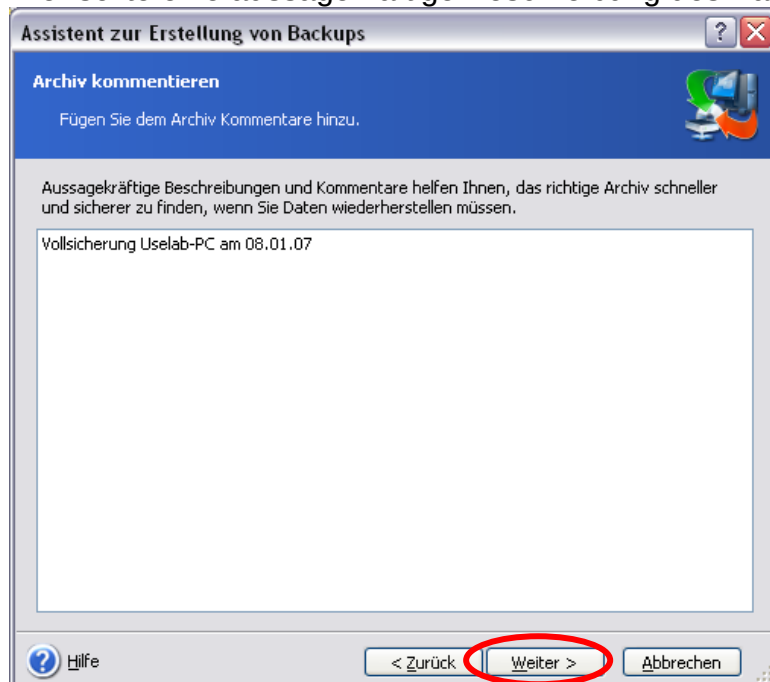


Abb. B4- 9: Beschreibung des Backups

## 9. Fertigstellen

Zur Sicherheit nochmal die angegebenen Informationen nachprüfen. Dann „Fertigstellen“ auswählen und den Backup-Vorgang starten (Abbildung B4 – 10). Dies kann jetzt einige Zeit dauern.

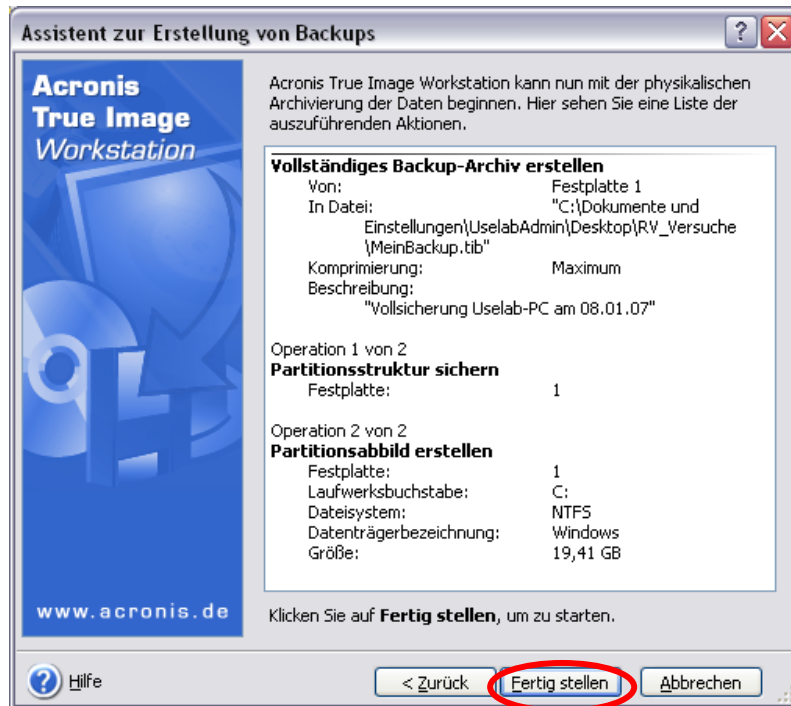


Abb. B4- 10: Starten des Backupvorgangs

### 3. Daten wiederherstellen

1. Bei Angabe der Aktionen „Wiederherstellung“ auswählen, siehe Abbildung B4 – 11.

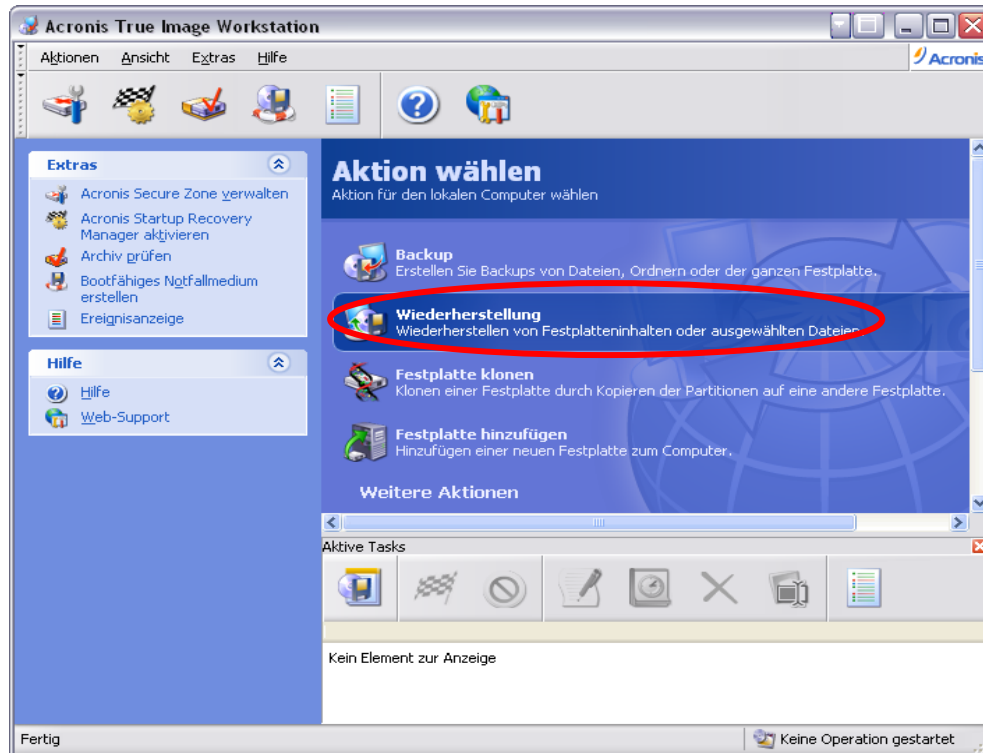


Abb. B4- 11: Festplatte soll wiederhergestellt werden

2. Der Assistent zur Wiederherstellung meldet sich. Im weiteren Verlauf den Anweisungen dieses Assistenten folgen.

## B5 Mousetrace

Der Mousetracer kann über die ausführbare JAR – Datei mousetrace.jar bequem vom Desktop des Rechners aus gestartet werden. Nach dem Start erscheint eine Benutzeroberfläche, wie sie in Abbildung B5 – 1 zu sehen ist.

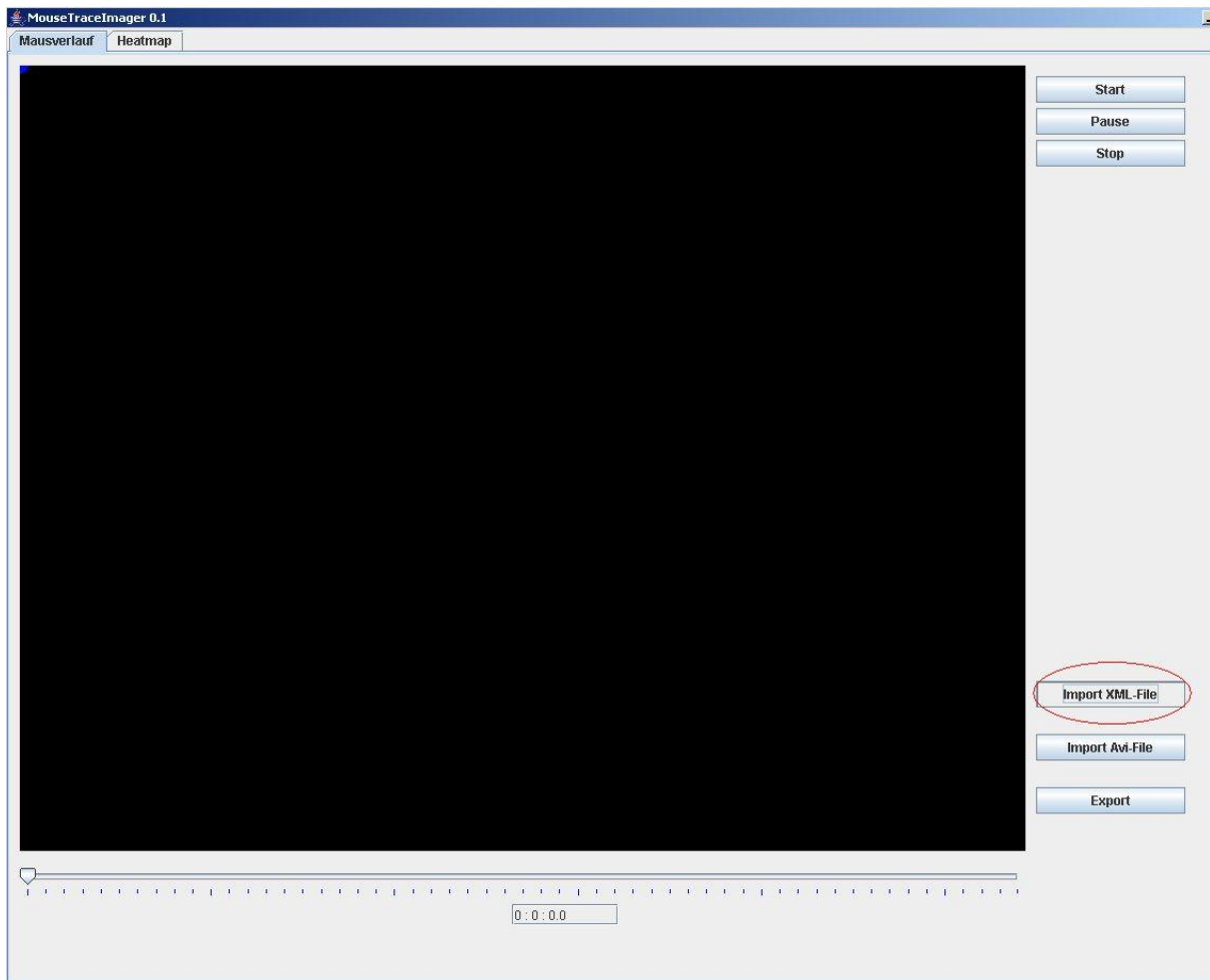


Abb. B5- 1: Mousetracer nach dem Starten

Nach dem Start hat man nun verschiedene Möglichkeiten zur Einstellung des Mousetracers. Zunächst einmal lässt sich über den Button „Import XML – File“ eine XML – Datei des Mousetrackers einlesen (siehe rote Markierung in der obigen Abbildung). Dies ist zwingend notwendig, wenn ein neuer Film über einen Mausverlauf dargestellt werden soll. Nach dem Laden wird bereits das erste Bild des Mausverlaufs angezeigt.

Über den Button „Import Avi – File“ lässt sich zusätzlich ein Hintergrundvideo (z.B. die zu dem Test gehörende Aufzeichnung des Eyetrackers) laden, das parallel während der Anzeige des Mausverlaufs mit angezeigt wird (Abbildung B5 – 2). Dies ist jedoch nicht notwendig, es genügt, die erforderliche XML – Datei einzulesen. Gleichzeitig bietet dieser Button die Möglichkeit, ein bereits vorhandenes Video des Moustracers zur Anzeige zu laden.

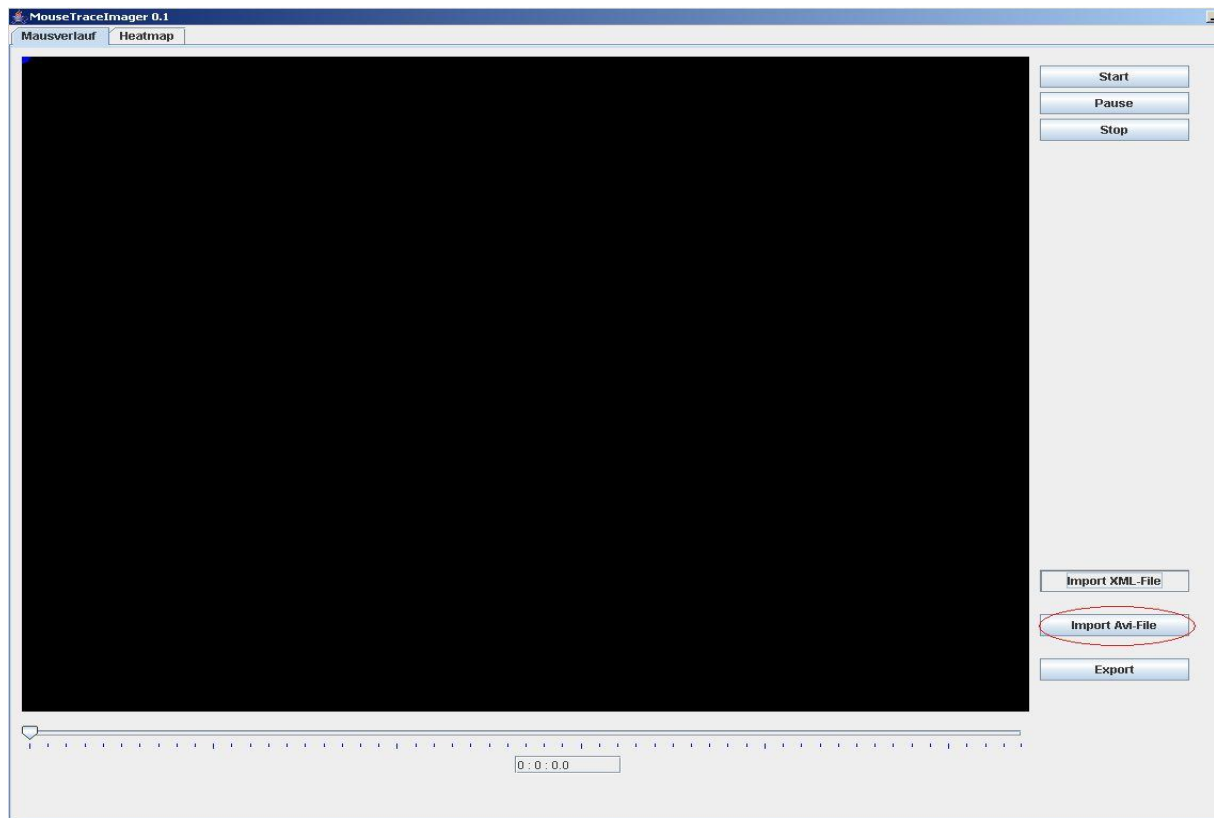


Abb. B5- 2: Importieren eines Hintergrundvideos

Über den „Export – Button“ können die Einzelbilder des Mausverlaufs als zusammengeschnittene AVI – Datei exportiert werden (Abbildung B5 – 3).

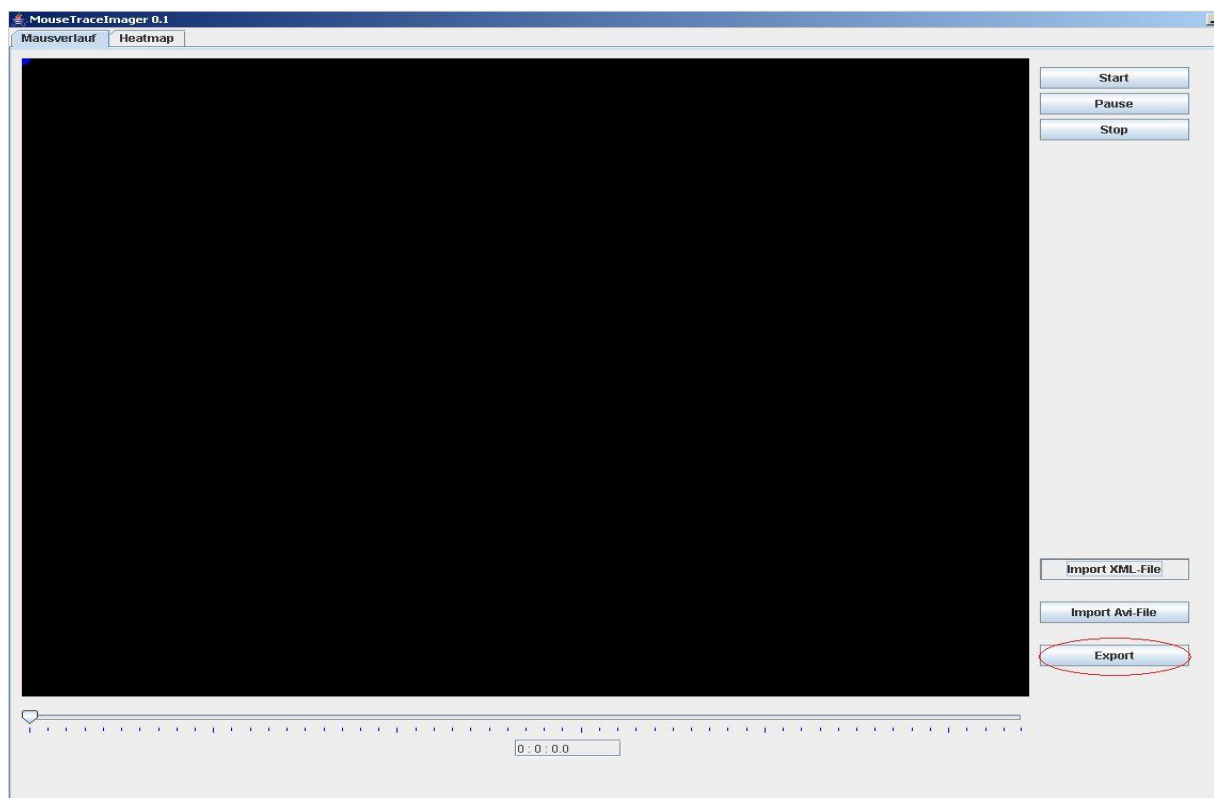


Abb. B5- 3: Exportieren eines Mausverlaufs

Der Fortschritt über den Mausverlauf lässt jederzeit über die Fortschrittsanzeige unterhalb des Bildbereichs anzeigen. Die verstrichene Zeit wird dabei in der kleinen Textbox festgehalten (siehe Abbildung B5 – 4).

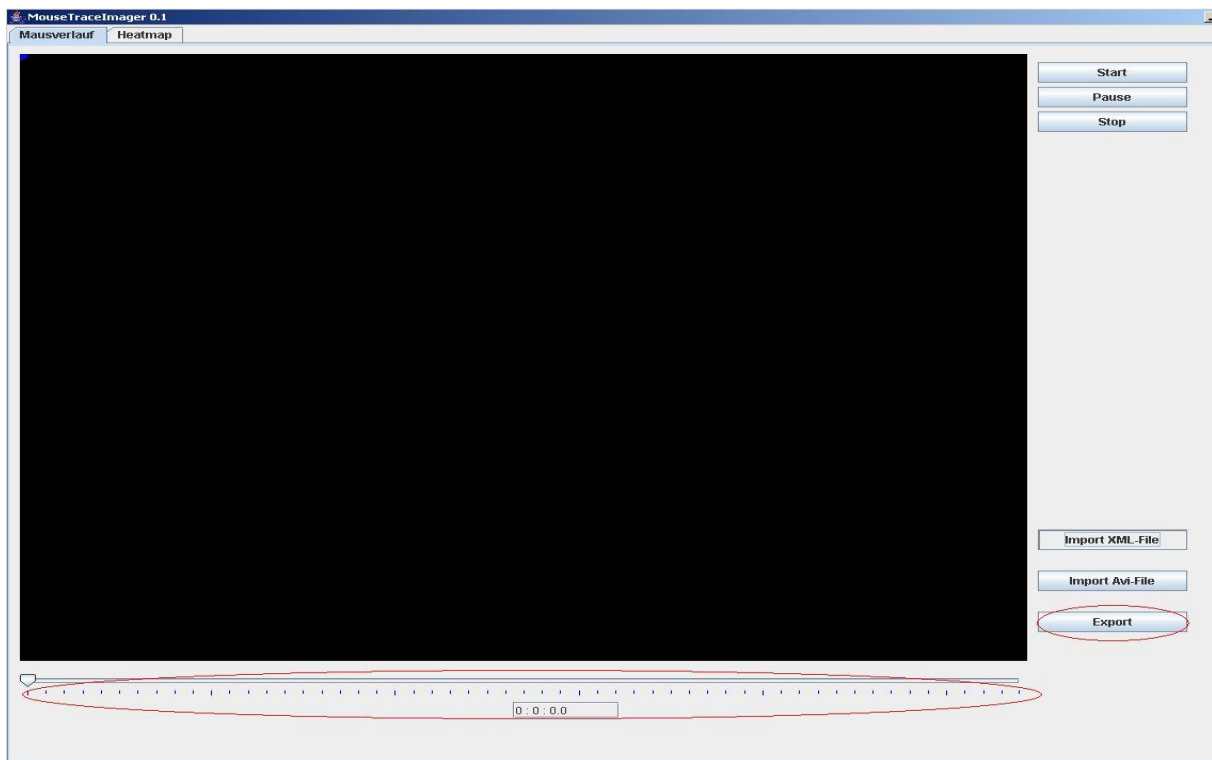


Abb. B5- 4: Fortschrittsanzeige

Zum Starten der Mausanzeige muss nun als nächstes der „Start“-Button betätigt werden (Abbildung B5 – 5).

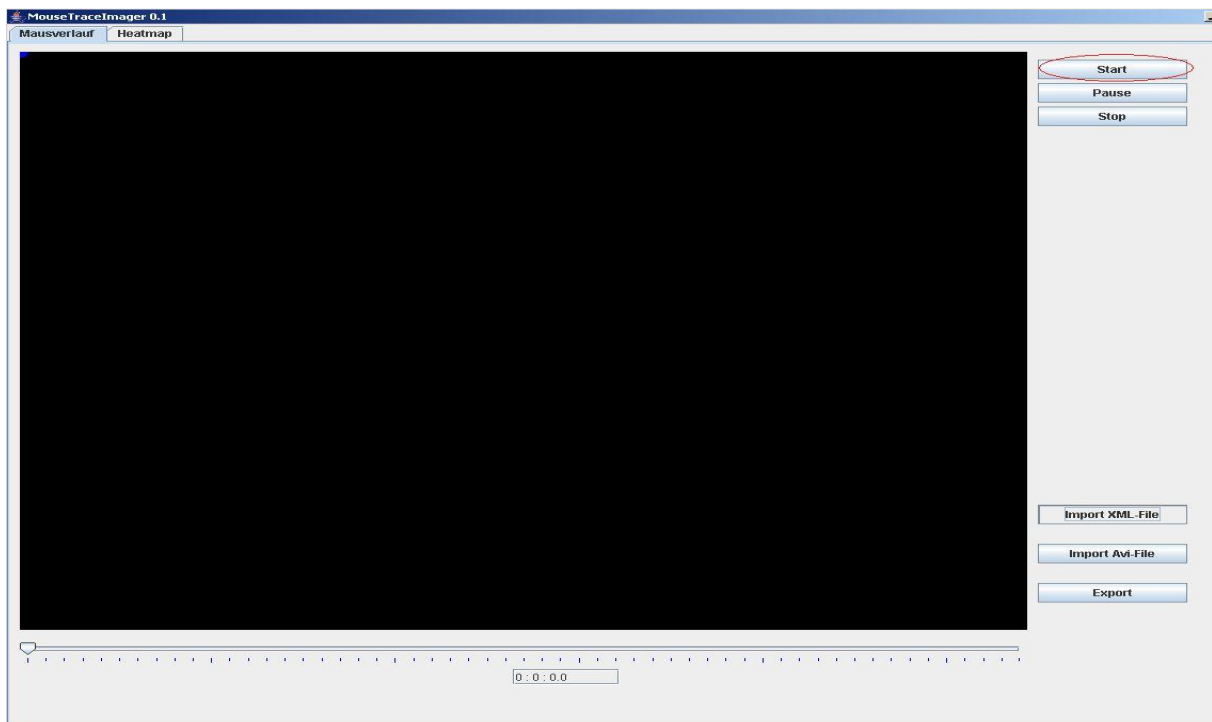


Abb. B5- 5: Start-Funktion in der Videoansicht

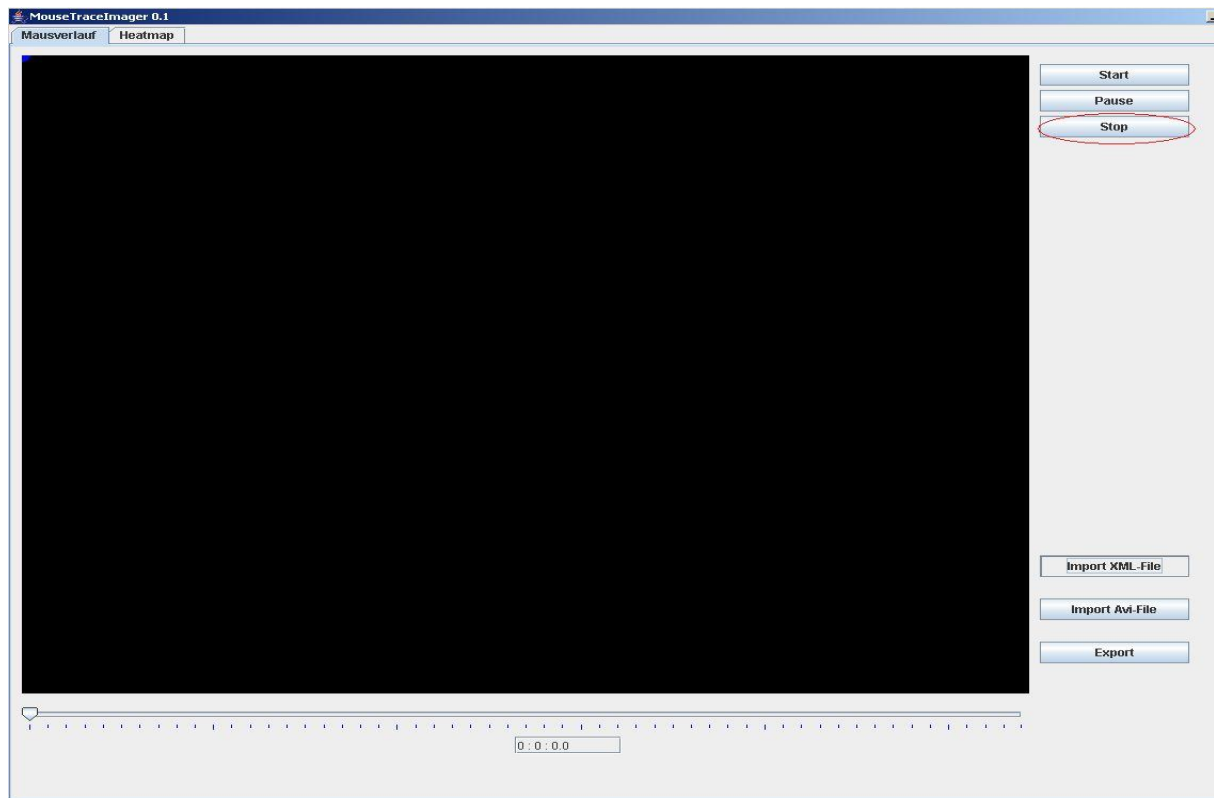


*Abb. B5- 6: Stop-Funktion in der Videoansicht*

Das Video über den Mausverlauf kann zu jeder Zeit über den „Stop“ – Button (Abbildung B5 – 6) oder über den „Pause“ – Button (Abbildung B5 – 7) angehalten werden.

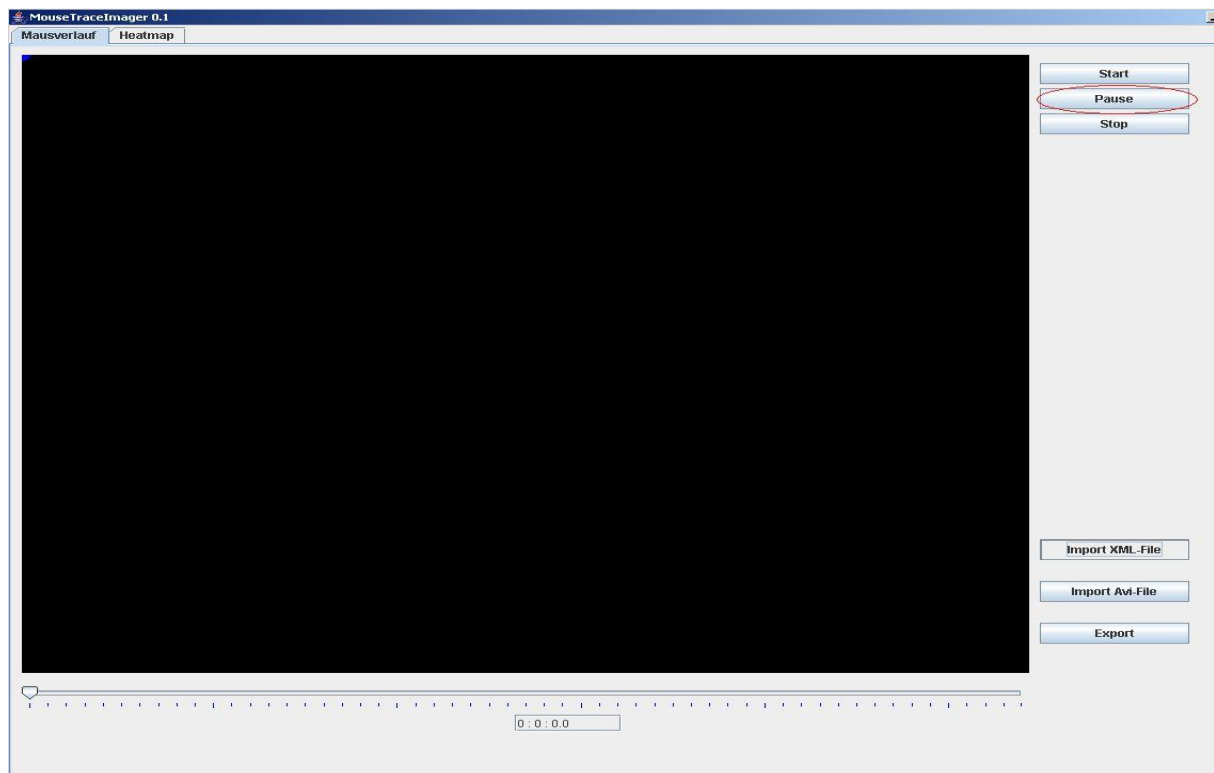


*Abb. B5- 7: Pause-Funktion in der Videoansicht*

Wird die Anzeige mittels „Pause“ angehalten, so kann die Aufzeichnung mit einem erneuten Klick auf „Pause“ von der letzten Stelle aus fortgesetzt werden. Wird die Anzeige dagegen mittels „Stop“ angehalten, so muss das Video neu gestartet werden.

## Heatmap – Ansicht

Über den Tab „Heatmap“ gelangt man in die Heatmap – Ansicht. Hier hat man die Möglichkeit, Ballungsgebiete des Mauseufenthalts zu bestimmten Zeitpunkten zu ermitteln, die in Form von Wolken dargestellt werden. Über die beiden Slider lässt sich dabei das betrachtete Zeitfenster einstellen. Mit dem oberen Slider wird der früheste Zeitpunkt, mit dem unteren Slider der späteste Zeitpunkt eingestellt (Abbildung B5 – 8).

**Achtung:** Ist keine XML – Datei eingelesen worden, so kann auch keine Heatmap gezeichnet werden. In diesem Fall sind die Slider nicht verstellbar. Es ist ebenfalls nicht möglich, den oberen Slider auf eine Position hinter dem unteren Slider zu setzen (der früheste Zeitpunkt kann nicht hinter dem spätesten Zeitpunkt liegen).

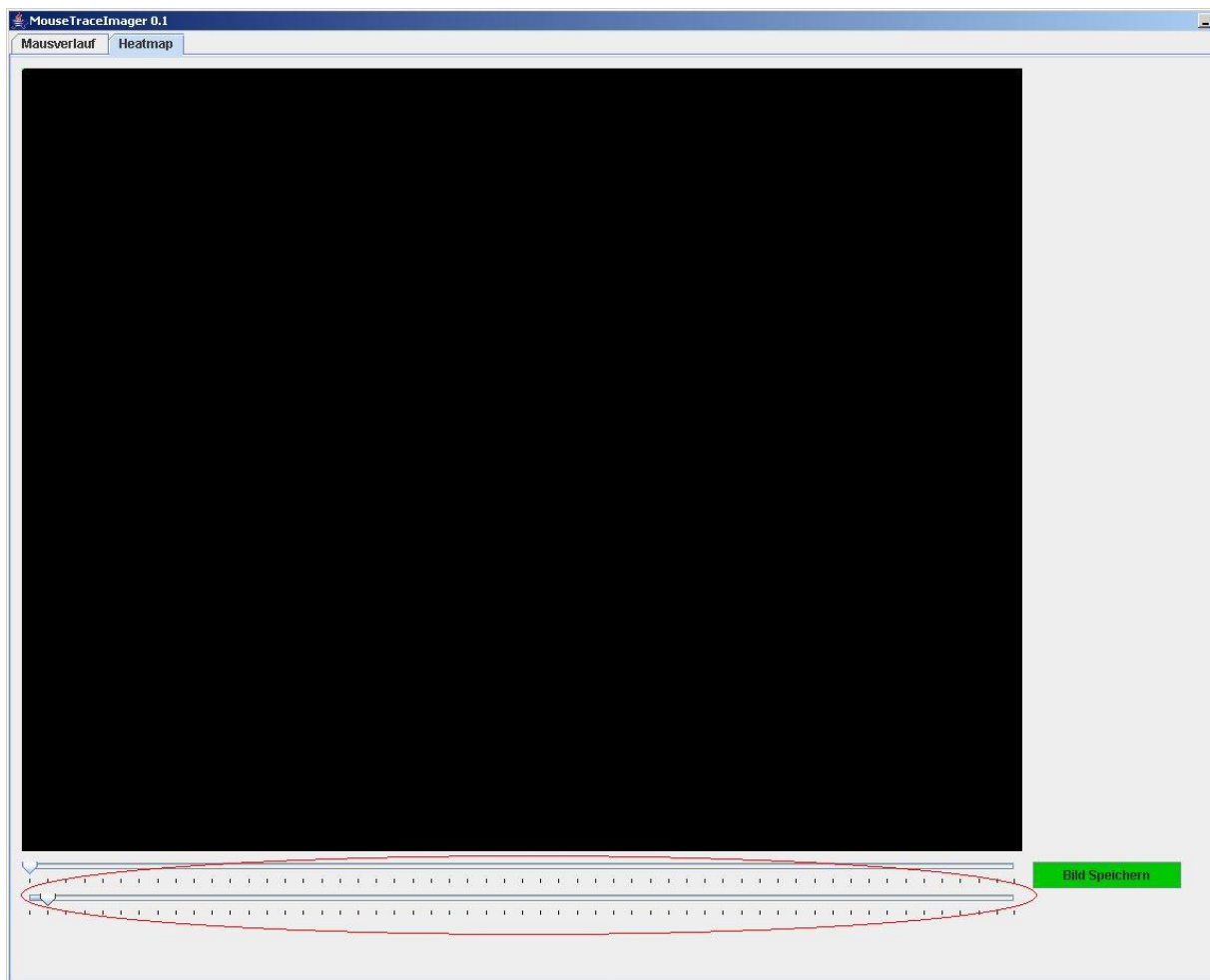


Abb. B5- 8: Slider in der Heatmap-Ansicht

Zusätzlich kann jedes einzeln angezeigte Bild als PNG – Datei abgespeichert werden. Dies ist über den Button „Bild speichern“ möglich (Abbildung B5 – 9).

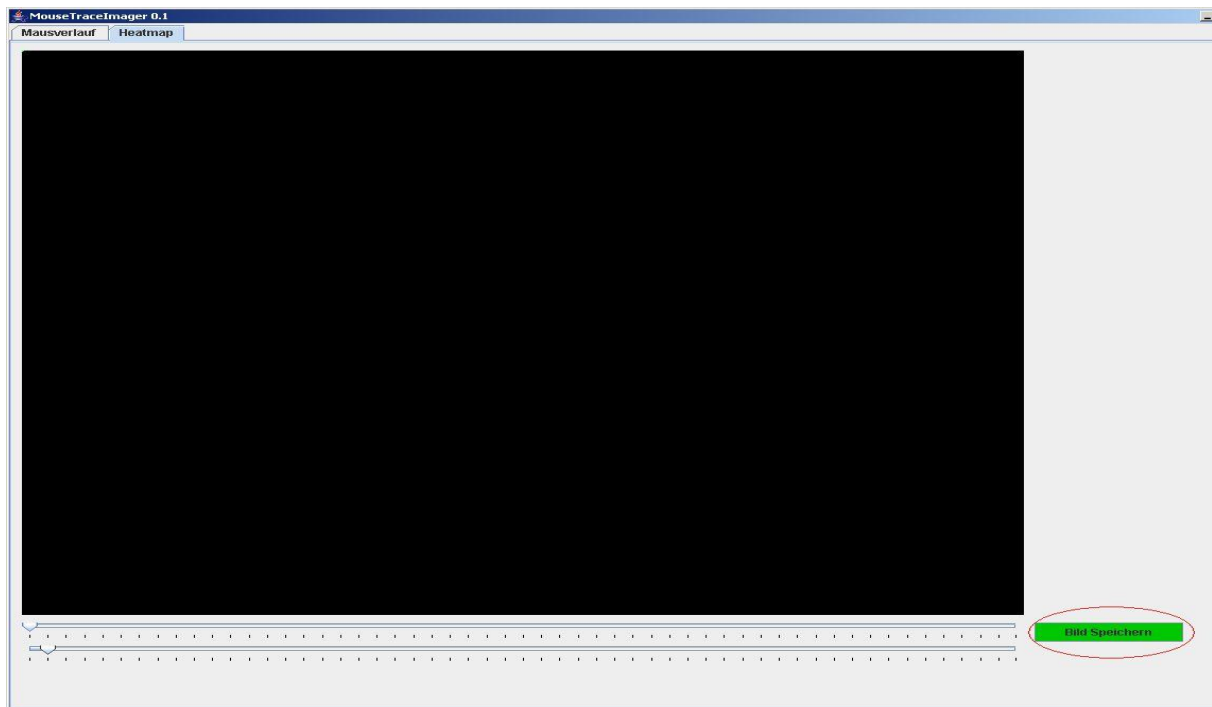


Abb. B5- 9: Speicherfunktion in der Heatmap-Ansicht

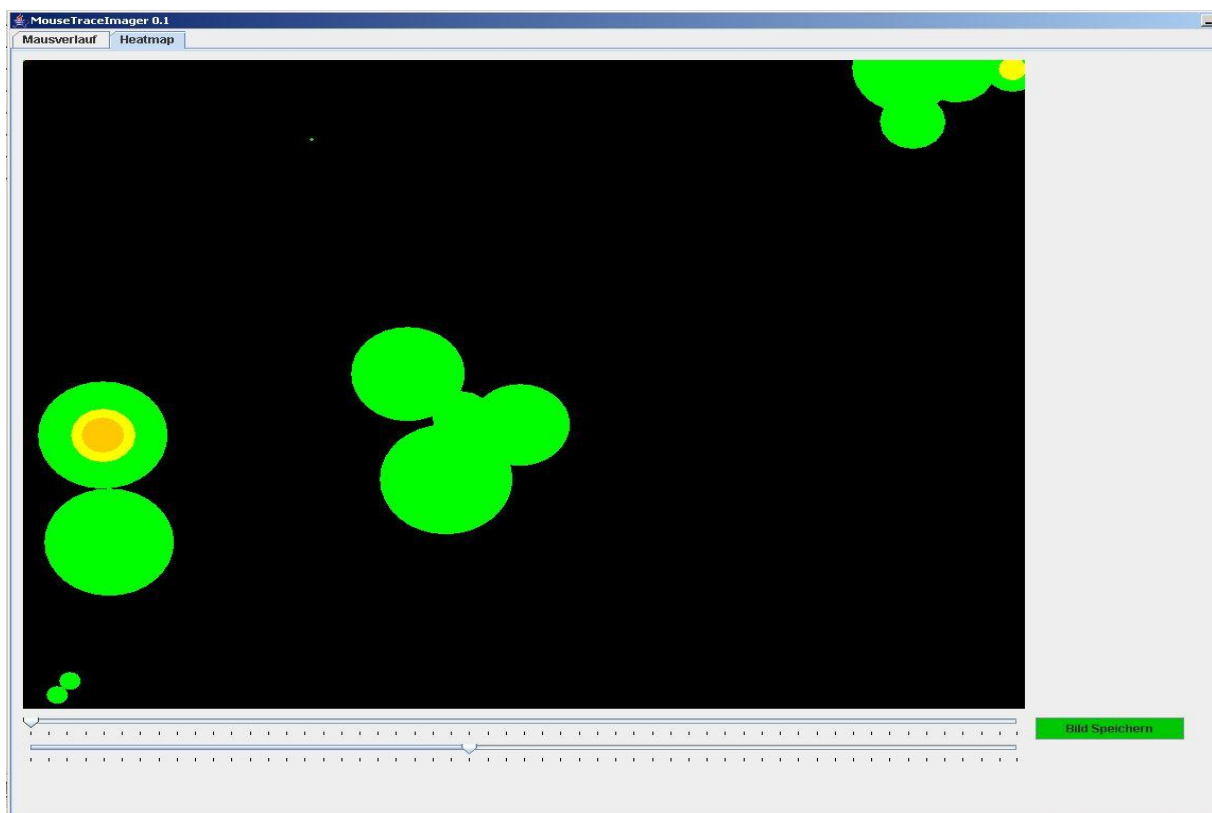


Abb. B5- 10: Heatmap im Mousetracer

Bei ordnungsgemässer Handhabung sollte eine Heatmap schliesslich etwa wie in Abbildung B5 – 10 dargestellt werden.

## B6 Bedienung des Gesamtsystems

### 1. Das System hochfahren

Im Testraum:

- Testrechner mitsamt Soundsystem einschalten und hochfahren. Welcher der beiden Testrechner verwendet wird, hängt vom Untersuchungsgegenstand ab. Passwörter notfalls beim Professor oder zuständigen Mitarbeiter erfragen. Software von Buzzertool und Mousetracker werden automatisch gestartet. Das Mousetracker – Fenster rechts aus dem Bildschirmbereich ziehen.
- Eyetracker einschalten, falls Verwendung erwünscht ist. Gleichzeitig die Software des Eyetrackers starten (Tobii Studio) und einrichten. Nähere Einzelheiten hierzu sind der Diplomarbeit von Stefan Richter zu entnehmen.
- Kameras mit der Fernbedienung aus kürzester Entfernung einschalten. Anmerkung: Die Kameras 1 bis 5 reagieren auf den Kanal 1, die Kamera 6 hingegen auf den 2.Kanal.

Im Regieraum:

- Schnittrechner (im Turm querliegend) und Capturerechner (über dem Schnittrechner) in der Reihenfolge von unten nach oben einschalten und hochfahren. Passwörter notfalls beim Professor oder zuständigen Mitarbeiter erfragen. Monitore einschalten.
- Falls benötigt, Linkstation einschalten (ganz oben im Rechnerturm)
- Verbindung Schnittrechner → externe Festplatte G: herstellen durch Öffnen des G:\ – Verzeichnisses auf dem Schnittrechner.
- Gleichen Vorgang auf den Capturerechnern wiederholen. Dies ist insbesondere für die Videoaufzeichnung von Bedeutung, da im Falle einer nicht korrekten Verbindung keine Speicherung der Kamerastreams auf der externen Festplatte erfolgt !  
Hinweis: Über die linke oder rechte STRG – Taste kann zwischen den einzelnen Rechnern hin – und hergewechselt werden.
- Auf beiden Capturerechnern das Aufzeichnungsprogramm Focus Capture Suite öffnen
- Metriken/Video – Switch einschalten
- Audio System Pioneer einschalten und Lautstärke auf - 31 dB einstellen
- Ton einschalten durch Einstecken des Steckers an der Mikrofonsteuerung
- Mikrofonsteuerung: Regler auf Mittelstellung bringen
- VGA – Bildschirm sowie Metriken – Bildschirm einschalten
- VGA – Steuerung einschalten
- Alle 6 Kamera – Monitore einschalten
- Positionen der Kameras über Kamerasteuerung einstellen  
Hinweis: Grundposition ist auf Taste 1 gespeichert, Position der Nachbesprechung auf Taste 2 der Kamerasteuerung

Wenn alles hochgefahren ist, sollte im nächsten Schritt ein Soundcheck durchgeführt werden. Es ist zu prüfen, ob Mikrofone und Lautsprecher sowohl im Testraum als auch im Regieraum ihre Funktionalität erfüllen.

Nachbedingungen:

- Ton ist in beiden Räumen hörbar
- Die Kameramonitore zeigen das Bild der jeweiligen Kamera
- Die Kamerabilder im Focus Capture Suite auf den Capturerechner sind identisch mit den Bildern von den Kameramonitoren und frei von Übertragungsstörungen
- Die Kameras ändern ihre Position entsprechend den Steuerungen über die Kamerafernbedienung
- Auf dem VGA – Bildschirm im Regiebild ist das exakte Abbild des Bildschirms vom Testrechner zu sehen, ebenso im Focus Capture Suite.  
Hinweis zur Software Focus Capture Suite:  
Auf dem 1. Capturerechner links werden die Kameras 1 – 4 angezeigt, in der Reihenfolge von rechts nach links und oben nach unten (also Kamera 1 oben links, Kamera 4 unten rechts). Auf dem 2. Capturerechner rechts werden auf der rechten Seite die Kameras 5 (oben) und 6 (unten) angezeigt, das VGA – Bild sollte auf der unteren linken Seite zu sehen sein. Der Quadrant oben links wird nicht verwendet. Anhand der Buttons „Sync“ kann eingestellt werden, welche Kamerafilme aufgenommen werden sollen. „Inlay“ schaltet die Anzeige des jeweiligen Kamerabildes ein bzw. aus.
- Auf dem Metriken – Monitor ist die Anzeige des Mousecapturers zu sehen

## 2. Aufzeichnung eines Usability – Tests

Vor der Aufzeichnung sind folgende Programme auf dem Schnittrechner zu starten:

- Client des Mousecapturers  
Eine entsprechende Verknüpfung liegt auf dem Desktop bereit.  
Anschliessend den Client mit dem Server (Testrechner) über das Menü Verbindung verbinden.  
Eine ausführliche Bedienungsanleitung des Mousecapturers kann der zugehörigen Dokumentation entnommen werden
- Client des Buzzertools  
Entsprechende Verknüpfung liegt auf dem Desktop bereit.  
Anschliessend Einstellungen wie in der entsprechenden Anleitung vornehmen.
- Zum ferngesteuerten Ansprechen der Aufnahmesoftware von Focus auf den Capturerechnern die Batch-Datei RunAll (liegt auf dem Desktop) starten.  
Daraufhin werden zwei Benutzeroberflächen angezeigt, mit denen die Aufzeichnung, die von den Capturerechnern durchgeführt wird, gestartet oder gestoppt wird.  
**Achtung:** Zuvor muss auf den beiden Capturerechnern die angesprochene Software geöffnet sein, da sie sonst nicht vom Schnittrechner aus angesprochen werden kann !!!

Anschliessend folgende Einstellungen vornehmen:

1. In den Feldern „Filename“ jeweils einen Dateinamen angeben
2. Das Feld „Channel“ beider Oberflächen muss auf 0 eingestellt sein
3. Im Adressfeld jeweils IP – Adressen der Capturerechner eingeben,  
für den linken Capturerechner die 10.0.2.19, für den rechten die 10.0.2.21.  
Siehe auch Abbildung B6 – 1.

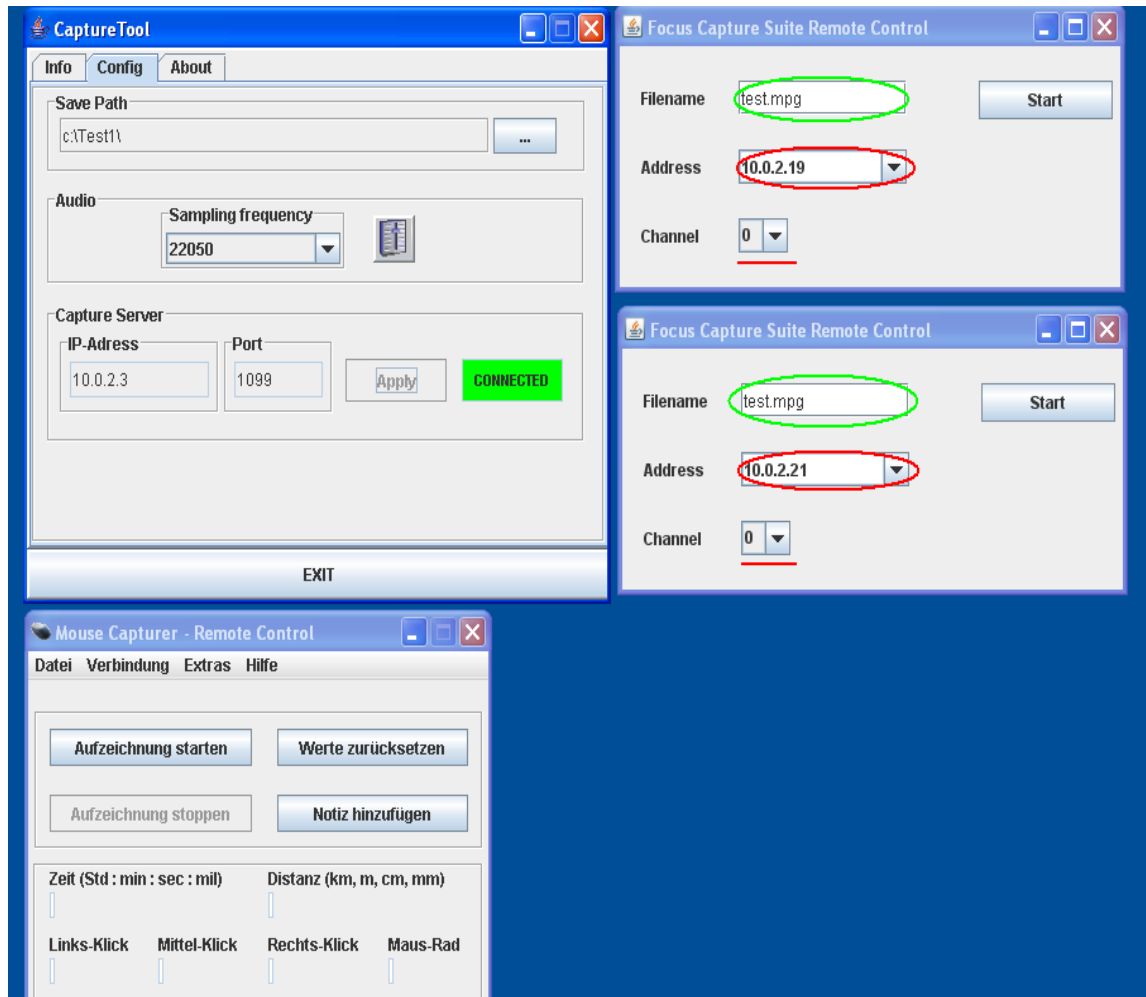


Abb. B6- 1: Voreinstellungen für die Videoaufzeichnung

Der Desktop des Schnittrechners sollte nun ungefähr der Abbildung B6 – 2 entsprechen.

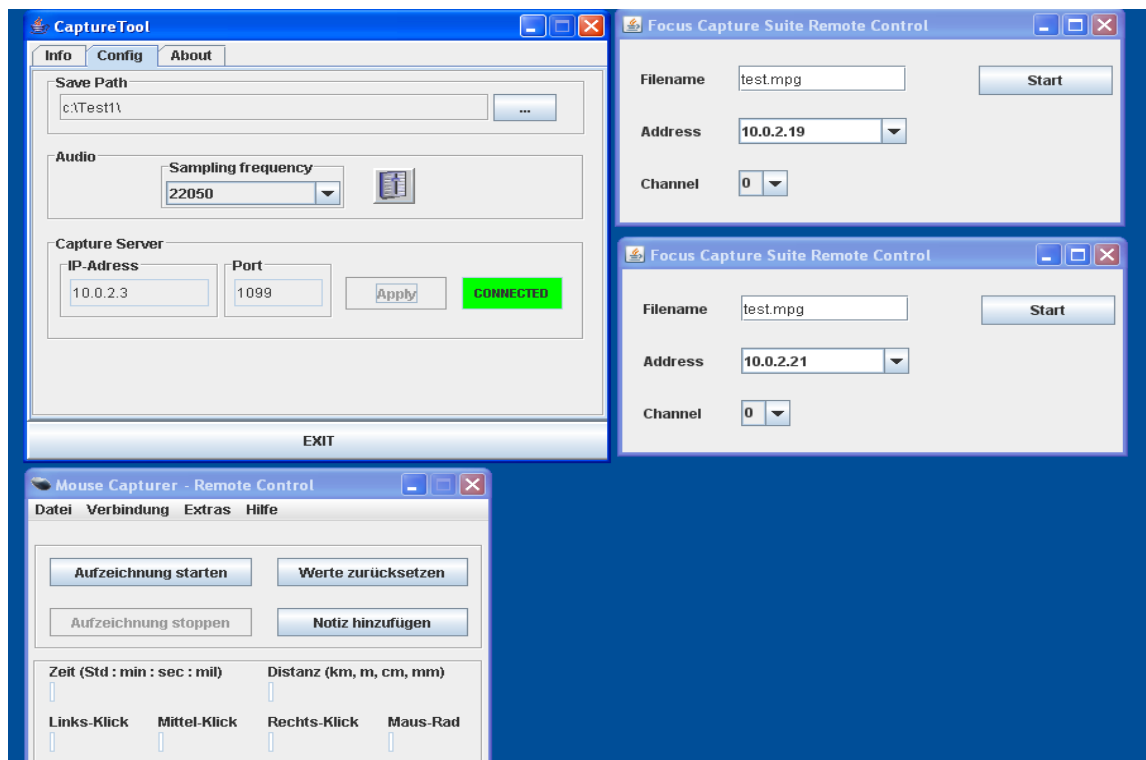


Abb. B6- 2: Benötigte Programme für die Aufzeichnung

Um die Aufzeichnung zu starten, sind nun die jeweiligen Startbuttons sowohl der beiden Fernsteuerungstools als auch des MouseCapturers innerhalb kürzester Zeit zu drücken (siehe Abbildung B6 – 3). Die Startbuttons der Fernsteuerungstools ändern daraufhin ihre Anzeige auf „Stop“.

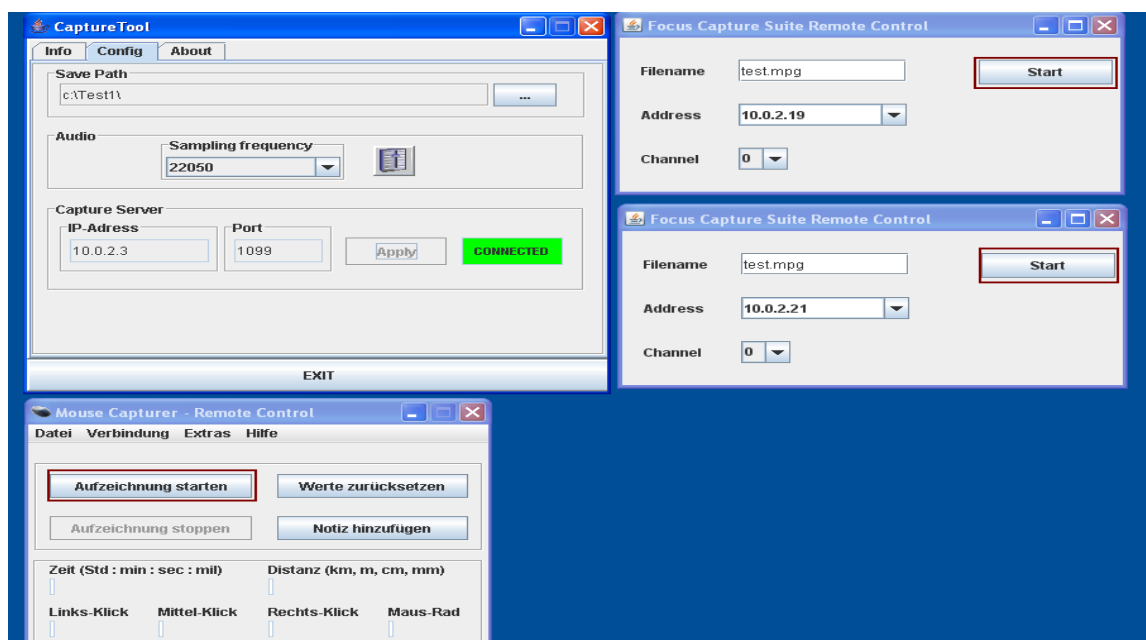


Abb. B6- 3: Aufzeichnung starten

Um die Aufzeichnung zu stoppen, sind die jeweiligen Stopbuttons sowohl der beiden Fernsteuerungstools als auch des MouseCapturers innerhalb kürzester Zeit zu drücken (siehe Abbildung B6 – 4). Die Startbuttons der Fernsteuerungstools ändern daraufhin ihre Anzeige zurück auf „Start“.

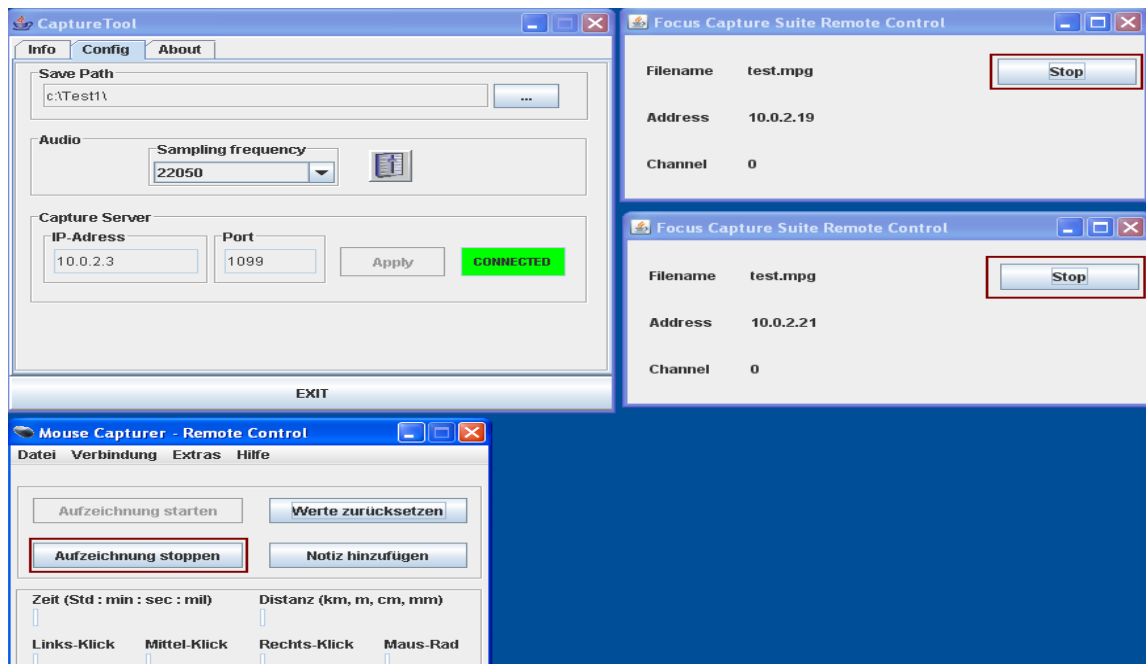


Abb. B6- 4: Aufzeichnung stoppen

### Achtung !!

Es ist unbedingt erforderlich, direkt auf beiden Capturerechnern nachzuprüfen, ob die Aufzeichnung auch wirklich angehalten wurde. Dies ist nicht immer der Fall. Erkennbar ist dies an dem Aufleuchten des roten Kreises in der Capture Suite (siehe auch grüne Markierung in der Abbildung B6 – 5). In solchen Situationen unbedingt den Button direkt daneben im 1. Quadranten manuell betätigen (siehe rote Markierung).

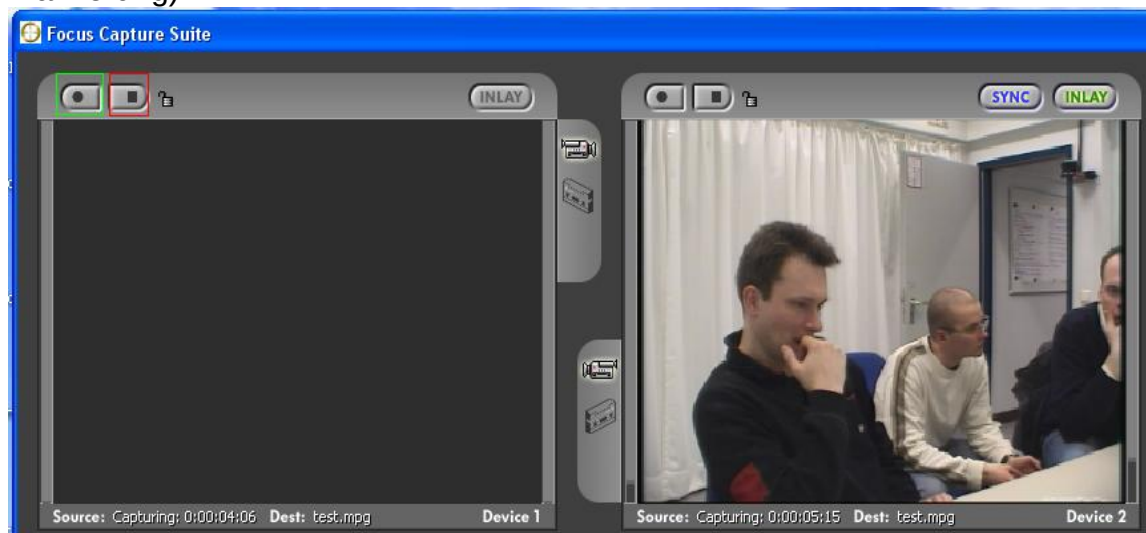


Abb. B6- 5: Focus Capture Suite auf dem Videograbber

Nach der Videoaufzeichnung sollten die Daten des Mousecapturers für die spätere Auswertung gesichert werden. Hierzu über das Menü „Datei“ gehen, den Punkt „Aufzeichnung exportieren (XML)“ wählen und die Datei in einem geeigneten Verzeichnis abspeichern (am besten in dem Ordner, der auch für die Daten des Buzzertools verwendet wird, dann hat man alles zusammen in einem Ordner). Anschliessend die Werte über den gleichnamigen Button auf 0 zurücksetzen.

### **3. Das System herunterfahren**

Um das System herunterzufahren, müssen einfach die Schritte aus dem Punkt 1. In umgekehrter Reihenfolge durchgeführt werden. Dabei sind folgende Abweichungen zu berücksichtigen:

- Die Linkstation ist erst dann auszuschalten, wenn die Archivierung vollständig durchgeführt wurde
- Wenn direkt im Anschluss an die Aufzeichnung die Nachbearbeitung der Videos erfolgt (schneiden, rendern), den Schnittrechner erst nach vollständiger Videobearbeitung herunterfahren. Sollte das Operatingtool benutzt werden, wird der Rechner automatisch heruntergefahren.

## B7 Any Video Converter

Für die Konvertierung der MPEG – Filme in ein anderes Format sind folgende Schritte auszuführen:

1. Über den Menüpunkt „Video hinzufügen“ jeden zu konvertierenden Film in den Converter laden. Informationen über den Film werden dann in der Liste angezeigt. Für jeden Film können zusätzlich Einstellungen wie Bildgrösse und Bildrate angepasst werden. Siehe auch entsprechende Markierungen in der folgenden Abbildung B7 – 1.

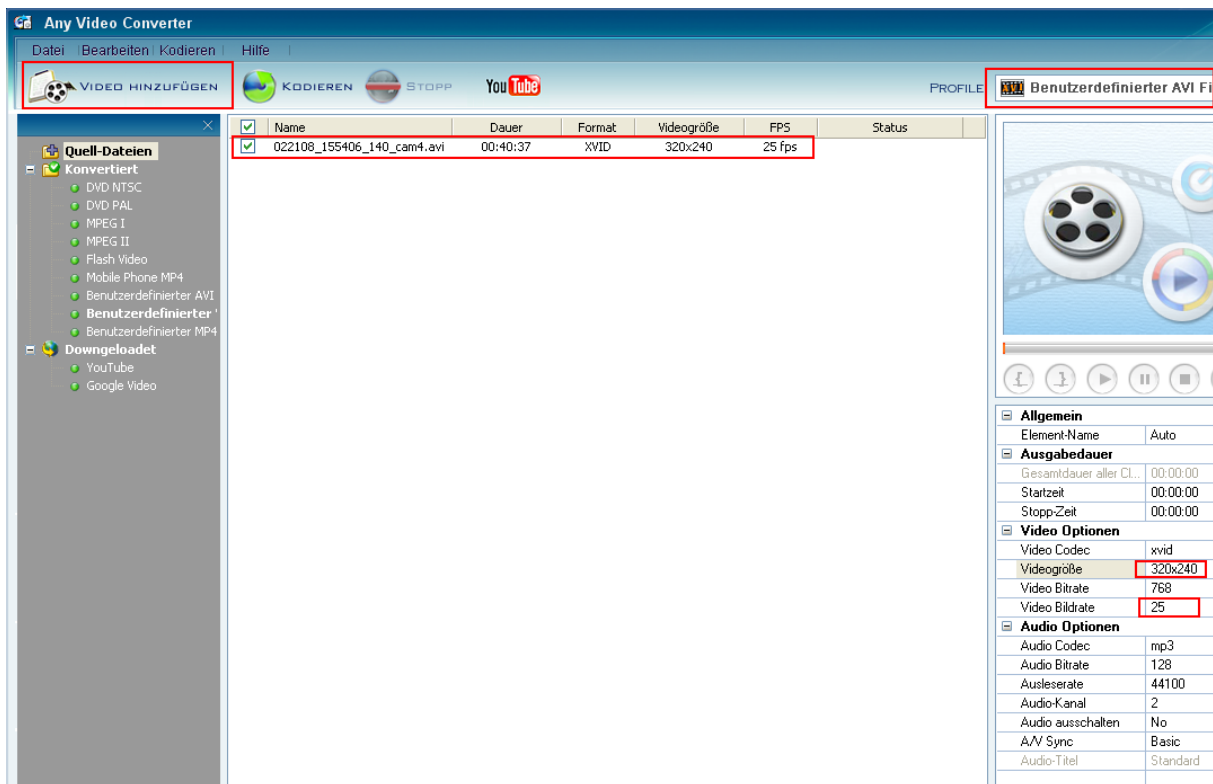


Abb. B7- 1: Einstellungen für Any Video Converter

2. Sind alle Dateien geladen, über das Auswahlfeld Profile das gewünschte Ausgabeformat (z.B. AVI oder WMV) festlegen, in das umgewandelt werden soll.
3. Über den Menüpunkt „Datei“ das Zielverzeichnis festlegen.
4. Über „Kodieren“ den Konvertierungsvorgang starten.

## Anhang C – Abbildungsverzeichnis

|                                                                              |     |
|------------------------------------------------------------------------------|-----|
| 1-1-1: Blick in den Testraum                                                 | 11  |
| 1-1-2: Blick in den Regieraum                                                | 11  |
| 1-2-1: Komponenten und Verbindungen im Usability – Labor vor Umbau           | 13  |
| 1-2-2: Netzwerkaufbau Usability – Labor bis September 2007                   | 15  |
| 1-2-3: Arbeitsplatz des Testleiters mit Kameramonitoren und Mikrofon         | 17  |
| 1-2-4: Komponenten und Verbindungen im Usability – Labor (aktuell)           | 18  |
| 1-2-5: Rechnerturm im Regieraum                                              | 20  |
| 1-2-6: Buzzertool mit VGA – und Metrikenmonitor                              | 20  |
| 2-3-1: Wunschvorstellung einer idealen Benutzeroberfläche                    | 27  |
| 3-2-1: Videoformate im Überblick                                             | 41  |
| 3-7-1: Tool von Daniel Wolf – Aktivitäten der linken Maustaste (Diagramm)    | 49  |
| 3-8-1: Komponenten des Operatingtools                                        | 52  |
| 3-9-1: Flexiblere Anzeige des Videoclips (Skizze)                            | 54  |
| 4-1-1: Komponenten des Mousetracers                                          | 55  |
| 4-1-2: Heatmap im Mousetracer                                                | 56  |
| 4-1-3: Klassen für die Ermittlung der relevanten Mausdaten                   | 57  |
| 4-1-4: Klassen für die graphische Darstellung                                | 60  |
| 4-1-5: Klassen der Benutzeroberfläche                                        | 62  |
| 4-1-6: Zusammenspiel der einzelnen Klassen                                   | 63  |
| 4-1-7: Mousetracer in Aktion                                                 | 64  |
| 4-2-1: Aufteilung der einzelnen Kameraansichten im fertigen Film (4:3)       | 66  |
| 4-2-2: Fertiger Film mit allen 6 Kameras und Eyetracker                      | 67  |
| 4-2-3: Abfolge bei der Skripterstellung (Sequenzdiagramm)                    | 69  |
| 4-3-1: Verzeichnisstruktur auf der Linkstation                               | 70  |
| 4-4-1: Operatingtool                                                         | 72  |
| 4-5-1: Buzzertool: Klassen auf der Serverseite                               | 74  |
| 4-5-2: Buzzertool: Klassen auf der Clientseite 1                             | 75  |
| 4-5-3: Buzzertool: Klassen auf der Clientseite 2                             | 76  |
| 5-2-1: Beispiel einer gespeicherten Heatmap                                  | 79  |
| B1- 1: Starten des Buzzerservers                                             | 97  |
| B1- 2: Starten des Buzzer-Clients                                            | 97  |
| B1- 3: Der Capture-Client konnte erfolgreich mit dem Server verbunden werden | 99  |
| B1- 4: Das Feld Info                                                         | 100 |
| B1- 5: Screenshot war erfolgreich                                            | 101 |
| B1- 6: Eine Aufnahme wurde gestartet                                         | 102 |
| B1- 7: Die Sprachaufzeichnung wurde angehalten                               | 103 |
| B1- 8: Schwarze und weisse Taste im Zusammenspiel – Ausgabe in der Anwendung | 105 |
| B1- 9: Die rote Taste wurde gedrückt und der aktuelle Test beendet           | 106 |
| B1- 10: Die Textdatei zu dem letzten Test                                    | 106 |

|                                                                           |     |
|---------------------------------------------------------------------------|-----|
| B2- 1: Der Uselabordner enthält ein Verzeichnis mit den AviSynth-Skripten | 120 |
| B2- 2: Karte laden                                                        | 121 |
| B2- 3: AviSynth-Skript in den Hauptbereich ziehen                         | 121 |
| B2- 4: Einstellungen vor dem Rendern                                      | 122 |
| B2- 5: Job Control von VirtualDub                                         | 123 |
| B2- 6: Anfangspunkt setzen                                                | 124 |
| B2- 7: Endpunkt setzen                                                    | 124 |
| <br>                                                                      |     |
| B3- 1: Funktionsübersicht des Operatingtools                              | 125 |
| B3- 2: AviSynth-Skripte erzeugen                                          | 126 |
| B3- 3: Filme automatisch über VirtualDub rendern lassen                   | 126 |
| B3- 4: Filme in das öffentliche Verzeichnis transferieren                 | 127 |
| B3- 5: Filme auf der Linkstation sichern                                  | 127 |
| B3- 6: Rechner runterfahren                                               | 128 |
| B3- 7: Komplettdurchlauf                                                  | 128 |
| <br>                                                                      |     |
| B4- 1: Acronis über Startleiste                                           | 129 |
| B4- 2: Auswahl des Backups                                                | 130 |
| B4- 3: Begrüssungsbildschirm des Assistenten                              | 130 |
| B4- 4: Art des Backups                                                    | 131 |
| B4- 5: Quellverzeichnis auswählen                                         | 131 |
| B4- 6: Zielpfad angeben                                                   | 132 |
| B4- 7: Auswahl des Archivtyps                                             | 132 |
| B4- 8: Auswahl der Optionen                                               | 133 |
| B4- 9: Beschreibung des Backups                                           | 133 |
| B4- 10: Starten des Backupvorgangs                                        | 134 |
| B4- 11: Festplatte soll wiederhergestellt werden                          | 135 |
| <br>                                                                      |     |
| B5-1: Mousetracer nach dem Starten                                        | 136 |
| B5-2: Importieren eines Hintergrundvideos                                 | 137 |
| B5-3: Exportieren eines Mausverlaufs                                      | 137 |
| B5-4: Fortschrittsanzeige                                                 | 138 |
| B5-5: Start-Funktion in der Videoansicht                                  | 138 |
| B5-6: Stop-Funktion in der Videoansicht                                   | 139 |
| B5-7: Pause-Funktion in der Videoansicht                                  | 139 |
| B5-8: Slider in der Heatmap-Ansicht                                       | 140 |
| B5-9: Speicherfunktion in der Heatmap-Ansicht                             | 141 |
| B5-10: Heatmap im Mousetracer                                             | 141 |
| <br>                                                                      |     |
| B6- 1: Voreinstellungen für die Videoaufzeichnung                         | 144 |
| B6- 2: Benötigte Programme für die Aufzeichnung                           | 145 |
| B6- 3: Aufzeichnung starten                                               | 145 |
| B6- 4: Aufzeichnung stoppen                                               | 146 |
| B6- 5: Focus Capture Suite auf dem Videograbber                           | 146 |
| <br>                                                                      |     |
| B7- 1: Einstellungen für Any Video Converter                              | 148 |

## D - Literaturverzeichnis

[Graf 2006] C. Graf: Emotionen in der Mensch-Maschine-Interaktion, Otto-von-Guericke-Universität Magdeburg, 2006

[Nelius 2003] M. Nelius: Mousemap-basierte Erkennung der Problemfelder von Anwendern bei der Bedienung von Software, Universität Rostock, 2003

[Nielsen 1994] Jacob Nielsen, Usability Engineering, Morgan Kaufman, New Edition, 1994

[Richter 2008] Stefan Richter, Bereitstellung eines Eyetracking-Systems für Lehre und Forschung, Diplomarbeit HAW Hamburg, 2008

[Schwarze 2003] André Schwarze: Konzept und Einrichtung eines Ergonomie-Labors, Diplomarbeit HAW Hamburg, 2003

[Ulas 2005] Bülent Ulas, Fatih Keles: Mouse Capturer Benutzerhandbuch, Arbeit im Projektpraktikum Usability-Engineering an der HAW Hamburg, 2005

[XviD 2006] C. Lampert, P. Massimino, M. Militzer, P. Ross: XviDCodec, <http://www.xvid.org>, 2006

[Wandmacher 1993] J. Wandmacher: Software-Ergonomie, Gruyter Verlag Berlin, 1993

[Wolf 2006] Daniel Wolf: Bereitstellung effizienter Auswertungsmethoden von Usability-Tests sowie prototypische Entwicklung eines Analysewerkzeuges, Bachelorarbeit HAW Hamburg, 2006

[Yaylacioglu 2007] Metin Yaylacioglu: Usability-Labor-Netzwerk (Netzwerkhandbuch), Arbeit im Projektpraktikum Usability-Engineering an der HAW Hamburg, 2007

[Zlotopolski 2006] Mark Zlotopolski, Tobias Wittmaack: BuzzerTool Dokumentation, Arbeit im Projektpraktikum Usability-Engineering an der HAW Hamburg, 2006

Zusätzlich: diverse Bedienungsanleitungen der neu beschaffenen Hardware für das Usability – Labor